计算机实用技术系列丛书(二)

视频多媒体
硬件技术与软件编程





学苑出版社

孙琮琦
闫德勤 编著

敬告用户

为满足广大用户对信息公路、多媒体、办公最佳工具 Excel 的快速学习和掌握，我们从美国著名电脑图书出版商之一 Alpha Books 公司引进一套傻瓜系列丛书，内容包括多媒体的安装、使用、制作；信息公路 Internet 的连入、访问、规则、功能；Excel 的安装、使用等技术。内容编排新颖，简洁明了，具有很强的可读性和实用性，是一套人见人爱、无师自通的必备工具系列丛书和最佳培训教材。

美国最畅销计算机傻瓜系列丛书

序号	书名	定价
1610	The Complete IDIOT'S Guide to Multimedia Multimedia 多媒体使用大全	25.00 元
1771	The Complete IDIOT'S Guide to USENET Newgroups 跟我加入 USENET Newgroups	17.80 元
1511	The Complete IDIOT'S Pocket Reference to the Internet: Internet 袖珍参考手册 Internet 入门指南	19.00 元
1772	The Complete IDIOT'S Guide to Mosaic 跟我学用 Mosaic	16.80 元
1702	The Complete IDIOT'S Pocket Guide to Excel 5 Excel 5 快速参考手册	9.00 元
1774	The Complete IDIOT'S Guide to Excel 跟我学用 Excel	16.50 元
1708	The Complete IDIOT'S Guide to Networking 网络入门指南	19.00 元

欲购以上图书的朋友请与 010－2541992，2562329 书刊部联系，
或传真 010－2579874，2561057 书刊部

ISBN 7－5077－0760－1/TP·7

本册定价：21.70 元

计算机实用技术系列丛书(二)

视频多媒体硬件技术
与 软 件 编 程

孙琮琦
闫德勤 编著
熊可宜 审校

学 苑 出 版 社

(京)新登字 151 号

内 容 简 介

本书主要介绍了 GRAND VIDEO 视频俘获卡的设计思想、硬件原理和应用软件编程的过程。另外，对 C 语言和汇编语言混合编程的 PCVIDEO 源程序也逐行进行了分析。

本书适用于开发多媒体应用或开发多媒体硬件与软件产品的技术人员，对于从事数字电视和图文电视的人也有一定的参考价值。

欲购本书的用户，请直接与北京 8721 信箱书刊部联系，邮政编码：100080，电话：2562329。

计算机实用技术系列丛书(二)

视频多媒体硬件技术与软件编程

编　　著：孙琮琦　闫德勤
审　　校：熊可宜
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：兰空印刷厂
开　　本：787×1092　1/16
印　　张：16.375　　**字数**：378 千字
印　　数：1～5000 册
版　　次：1995年10月北京第 1 版第 1 次
ISBN 7—5077—0760—1/TP・7
本册定价：21.70 元

学苑版图书印、装错误可随时退换

前　言

目前,计算机应用已进入多媒体时代,以计算机为中心进行图像、图形、声音和文字的高效输入、存储、处理、检索、通讯和输出是一种趋势。我国的多媒体应用也已起步。由于图像信息远比声音和可用代码表示的文字信息量大,特别是活动图像的输入、处理、传送和输出所需的技术和开发成本都比较大。它们成了普及多媒体技术应用的薄弱环节。GRAND VIDEO 视频俘获卡是台湾生产的比较大众化的产品。其它视频图像卡在电路上和它大同小异。它采用了数字电视技术,潜在的分辨率又比目前的电视的分辨率高。本书对它的设计思想,硬件原理和应用软件编程的全过程和每一细节进行了分析,希望对开发多媒体硬件和软件新产品、开发数字电视、高分辨电视和应用的读者会有所帮助。此外,本书对 C 语言和汇编语言混合编程的 PCVIDEO 源程序的结合硬件进行了逐行分析,对于迫切希望迅速提高 C 语言编程能力的读者也希望有所帮助。

本书第一章叙述 GRAND VIDEO 的安装和初步应用,第二到四章是与硬件原理有关的知识。因为 GRAND VIDEO 卡必须和 VGA 兼容的显视卡配合使用,作为预备知识,我们也介绍了目前在我国使用很广、性能价格比较高并且具有 1024×768 256 色的 TVGA8900 卡的内部寄存器及高级编程的有关技巧,作为作者另一本书《TVGA 卡应用开发编程入门》的补充。第四章介绍的电路图是我们对 GRAND VIDEO 卡实测的结果。第五到七章是 DOS 和 WINDOWS 两用库函数的全面和深入细致的分析。第七章对 C 和汇编源程序的逐条分析加中文注释,重点放在程序与硬件密切相关的核心部分。与硬件无关的部分,可能在别的书籍中出现的内容,例如图像变换则被省略。第八章介绍 PCVIDEO 的 DOS 库函数 PCVIDEO 的生成方法和在中文 UCDOS 3.0 环境下应用编程方法。

目　　录

第一章　安装和使用 Grand Video 卡 …… 1
第二章　一些预备知识 …… 14
　2.1　TVGA 属性控制器 …… 25
　2.2　I2C 总线 …… 26
　2.3　判优和时钟产生 …… 30
　2.4　7 位地址格式 …… 33
　2.5　I2C 总线的电气特征 …… 34
　2.6　10 位寻址 …… 35
第三章　有关集成电路 …… 37
　3.1　82C9001 PC 视频窗控制器说明 …… 37
　3.2　寄存器 …… 44
　3.3　视频模拟输入接口 TDA8708A …… 52
　3.4　SAA9051 数字多制式电视译码器 …… 55
　3.5　数字电视系统时钟信号发生电路 SAA9057A …… 71
　3.6　视频控制和自动截止白电平控制 TDA4680 …… 72
　3.7　数模变换视频处理器 SAA9060 …… 73
第四章　GRAND VIDEO 硬件原理 …… 76
第五章　PCVIDEO DOS/WINDOWS 两用库函数 …… 84
第六章　PCVIDEO 软件所用内包文件和头文件 …… 104
第七章　PCVIDEO 源程序分析 …… 126
　7.1　初始化子程序，结束子程序 …… 126
　7.2　视频俘获，标定裁剪和显示子程序 …… 145
　7.3　关于活动视频图像存储和重装的子程序 …… 158
　7.4　直接寄存器操作子程序 …… 190
第八章　应用编程 …… 219

第一章　安装和使用 Grand Video 卡

GRAND VIDEO 卡和原计算机中的图形显示卡(VGA,TVGA 或其它 SUPER VGA)合并使用。先把它插在一空槽中,然后用随卡 26 线扁平特征电缆联结卡上方 P2 到 VGA 或其它图形适配卡的上方的数字图形特征插座,拔下显示器联到 VGA 的电缆,改联到 GRAND VIDEO 的 P3 15 孔插座(背后上方),最后把随卡的另一根电缆的带三个屏蔽电缆插头的那一端插到 GRAND VIDEO 卡的 P4 15 孔插座(背后下方),电缆另一端接到 VGA 卡原先接显示器的那端。最后,把视频图像信号源,例如录象机,放象机,电视机或 AV 的视频输出经屏蔽电缆联接到 GRAND VIDEO 的红色视频插头上。这样我们就完成了 GRAND VIDEO 卡的硬件安装工作。

GRAND VIDEO 卡的应用软件有的是在 DOS 下工作的,有的是在 WINDOWS 下工作的。以下介绍随卡附送的几种简单软件工作方法:

1. EXAMPLE

EXAMPLE. EXE 是可以在 DOS 环境运行的,不带参数的程序。它的 C 语言源程序的汉化版将在第八章分析。

2. VIDEOLAN. EXE 是北京大恒图像视觉有限公司的产品。在 DOS 下工作,有中文菜单及窗口,操作简便。

3. WINVIDEO

WINVIDEO. EXE 是在 WINDOWS 3.1 环境下运行的示范程序。在中文 WINDOWS 环境下可以按中文提示工作。它的源程序也随卡提供。

4. DEMO

PCV_DEM. EXE 是一种 DOS 环境下的命令解释器,它一共可解释 29 条命令,它的使用方式为:

PCV_DEM 便笺文件名

便笺文件类似于批量文件,但其中的每一条命令不是 DOS 命令,而是以下 29 条命令之

1. DEFBGND

语法

DEFBGND n.

此命令定义某一颜色作为整个 VGA 显示的底色,用于某些图形操作命令,例如后面将要介绍的 GMOVE 及 GANIMATE。执行此命令将 VGA 存储区用所定义的颜色值填充。

参数	类型	描述
n	十进制数	n 为 VGA 模式 12 所定义的十六种颜色之一,对于此值实际代表的颜色,参看命令 EXTPAL 的解释。

2. FITNSHOW

语法

FITNSHOW x1 y1 wid hgt stpx stpy stpw stph count delay

此命令用于将显示的活动视频图像窗口的大小及位置按预定义的方向从一处移到另一处。外部视频信号总是充满显示的整个窗口。运行此命令即可以看到不断变换窗口大小的活动图像的显示效果。

参数	类型	描述
x1,y1	十进制数	此两参数定义了窗口初始时左上角在 VGA 屏幕上的位置。
wid,hgt	十进制数	此两参数定义了窗口初始时的宽度(wid)和高度(hgt)。
stpx,stpy	十进制数	此两参数定义了窗口左上角坐标相对于上次坐标的位移量,负数表示坐标值减小,正数表示坐标值增加。 即：Newx＝oldx＋stpx Newy＝oldy＋stpy
stpw,stph	十进制数	窗口的宽度(stpw)及高度(stph)相对于上次窗口宽度及高度的变化。负数表示减小,正数表示增加。 即：Neww＝oldw＋stpw Newh＝oldh＋stph
count	十进制数	此值定义了窗口大小及位置变化的次数。
delay	十进制数	此值定义了两次窗口的变化之间的间隔时间。 间隔＝(delay)×1/60 秒

3. FITVIDEO

语法

FITVIDEO x1 y1 wid hgt

此命令将整个外部视频信号显示于此命令所定义的视频显示窗口内。

参数	类型	描述
x1,y1	十进制数	此两参数为定义窗口的左上角坐标
wid,hgt	十进制数	此两参数为定义窗口的宽度(wid)和高度(hgt)

4. FREEZE

语法

FREEZE

此命令冻结当前在活动窗口内显示的图像,也就是说,视频显示区的图像不再被来自于外部的视频信号所修改。

5. GANIMATE

语法

GANIMATE x1 y1 wid hgt stpx stpy count delay mask

此命令实现将 VGA 上的一个矩形显示窗口连续的在 VGA 显示区从一处按预定方向移到另一处。此命令的适当使用会产生出动画效果。

参数	类型	描述
x1，y1	十进制数	在 VGA 上定义的矩形窗口的左上角初始坐标。
wid,hgt	十进制数	在 VGA 上定义的矩形窗口的初始宽度(wid)和高度(hgt)。
stpx,stpy	十进制数	此两参数定义了窗口左上角坐标相对于上次坐标的位移量,负数表示坐标值减小,正数表示坐标值增加。 即：Newx＝oldx＋stpx Newy＝oldy＋stpy
count	十进制数	窗口位置变化的次数。
delay	十进制数	定义每次窗口的变化的时间隔时间。 间隔＝(delay)×1/60 秒
mask	十进制数	此值定义 VGA 显示模式 12 时的 4 位平面的某一位平面掩码值。它相对于 4 位平面从最高至最低的每一位。

6. GBOX

语法

GBOX x1 y1 wid hgt color mask

在 VGA 存储器内画一矩形实心框

参数	类型	描述
x1,y1	十进制数	方框在 VGA 显示区的左上角位置坐标
wid,hgt	十进制数	方框的宽度(wid)和高度(hgt)
color	十进制数	在 VGA 上的实心方框区的颜色。所对应 color 的实际颜色,参看命令 EXTPAL。
mask	十进制数	此值定义 VGA 显示模式 12 时 4 个 bit 面中的掩码 bit 平面。它相对于 4bit 平面从最高至最低的每一平面。

7. GCIRCLE

语法

GCIRCLE radius x1 y1 color stph count delay mask

在 VGA 显示存储器中画一系列具有相同圆心的实心圆,运行此命令就会得到一个不断变大的圆。

参数	类型	描述
radius	十进制数	VGA 上圆的初始化半径
x1,y1	十进制数	圆心在 VGA 的坐标
color	十进制数	在 VGA 上的实心圆区的颜色。所对应 color 的实际颜色,参看命令 EXTPAL。
stph	十进制数	圆半径相对于上次半径的变化,负数为减少,正数为增大。

		即：新半径＝老半径＋stpx(或 stpy)
count	十进制数	VGA 圆的半径变化的次数。
delay	十进制数	圆的半径每次发生变化的时间隔时间。
mask	十进制数	此值定义 VGA 显示模式 12 时 4 个位面中的掩码位平面。它相对于 4 位平面从最高至最低的每一平面。

8. GCOPY

语法

GCOPY x1 y1 x2 y2 wid hgt offs

此命令将一个 VGA 上的矩形区从一个位置拷贝到另一个位置。

参数	类型	描述
x1,y1	十进制数	原始矩形区左上角在 VGA 上的坐标。
x2,y2	十进制数	目标矩形区左上角在 VGA 上的坐标。
wid,hgt	十进制数	VGA 上的方形区宽度(wid)和高度(hgt)。
offs	十进制数	此参数必须总为 128。

9. GLDVMP

语法

GLDVMP filename x1 y1

此命令将一个具有"VMP"格式的文件调入 VGA 显示存储区。"VMP"格式的文件定义将在后面加以解释。但在应用中一般均是利用提供的一个格式转换函数 PCX2VMP. EXE 将. PCX 格式的文件转为. VMP 格式的文件,因为可以产生. PCX 格式文件的创作工具实在是太多了。

参数	类型	描述
filename	ASCII	要调入 VGA 显示区的数据文件名。
x1,y1	十进制数	矩形区在视频卡显示区(1024×512)内的左上角坐标。

注意：如果调入的数据超出了 640×480 的范围,在 VGA 屏幕上就看不到显示。

10. GMOVE

语法

GMOVE x1 y1 x2 y2 wid hgt mask

此命令将 VGA 显示区内的一矩形区从原始位置移到目的位置,并将原始位置处的图像清除,添加由 DEFBGND 所定义的背景色

参数	类型	描述
x1,y1	十进制数	VGA 显示的原始矩形区的左上角坐标,范围在 640×480 之内。
x2,y2	十进制数	VGA 显示区内目的区的左上角坐标,范围在 640×480 之内。
wid,hgt	十进制数	VGA 显示的矩形区的宽度(wid)和高度(hgt)。
mask	十进制数	请见 GBOX 中关于 mask 参数的解释。

11. GSTAR5

语法

GSTAR hgt x1 y1 color stph count delay mask

在 VGA 显示区画一系列实心的五星状图形，它们具有相同的中心点，其结果看上去是屏幕上生长出了许多星星。

参数	类型	描述
hgt	十进制数	从五星的任一尖角处至中心的距离。
x1,y1	十进制数	VGA 上显示的五星中心点坐标，范围在 640×480 内。
color	十进制数	VGA 上显示的五星颜色，及此颜色所对应的在 VGA 上的颜色。参看命令 EXTPAL 中的定义。
stph	十进制数	当前 VGA 上的五星相对于上次显示的五星长度变化。负数表示缩小，正数表示放大。 即：Newradius＝oldradius＋stph
count	十进制数	VGA 上显示的五星长度将变化的次数。
delay	十进制数	两次变化的时间隔时间。间隔＝delay/60 秒。
mask	十进制数	参见 GBOX 中关于 mask 参数的解释。

12. INIT

语法

INIT

此命令将视频卡上的所有寄存器进行初始化，此后视频卡即可正常的工作。

13. JUMP

语法

JUMP jmplabel

此命令允许用户通过分支从一处跳到另一处改变命令序列实现无限循环，要转向的命令序列入口以命令 LABEL 标记(下面将给予解释)，它有相同的 jmplabel。

参数	类型	描述
jmplabel	ASCII	一个不超过 6 位的字符串。

14. LABEL

语法

LABEL jmplabel

此命令用于给出相应于 JUMP 命令中所要求的标号，以保证可以更改执行序列。

参数	类型	描述
jmplabel	ASCII	一个不超过 6 位的字符串。

15. NEWPAGE

语法

Newpage

此命令用于将 VGA 显示的内容变为新的一页，也就是说将清除视频卡上的活动图像窗口，并将 VGA 显示器以缺省的颜色白色(颜色号为 15)添充。因此 VGA 上将显示白色，缺省

的颜色可以通过 EXTPAL 命令加以改变。

16. PAUSE

语法

pause

此命令暂停命令解释器的工作，并在 PC 机的喇叭中发出短促的两声等待输入任何除 ESC 以外的键。当键入了任何非 ESC 以外的键时，命令解释器将重新启动并继续执行下面的命令。

17. QUIT

语法

QUIT

此命令退出命令解释器进入 DOS 提示，其作用与按下 ESC 相同。

18. RANDOMSTAR5

语法

RANDOMSTAR5 count

在 VGA 屏幕上画一系列随机大小，位置及颜色的实心五星。

参数	类型	描述
count	十进制数	VGA 上产生的随机五星的数目。

19. REM

语法

REM chrstring

此命令后的字符串以<RETURN>结束，命令解释器不对此串进行解释，而只作为增强命令文件的可读性使用，属解释性说明。

参数	类型	描述
chrstring	ASCII	用于解释性的描述字符串。

20. SHOWVIDEO

语法

SHOWVIDEO x1 y1 wid hgt stpx stpy stpw stph count delay

此命令按预定的方向连续改变视频卡显示的图像窗口的大小及位置，从 VGA 显示区一处至另一处。外部的视频图像并不充满所开窗口，除了外部视频图像不发生变倍外此命令的作用与命令"FITNSHOW"相同。运行此命令将产生出一个通过改变窗口大小实现外部视频图像在窗口滑动的显示效果。

参数	类型	描述
x1,y1	十进制数	初始窗口的左上角坐标。
wid,hgt	十进制数	初始窗口的宽度(wid)和高度(hgt)。
stpx,stpy	十进制数	窗口的左上角位置相对于上次窗口位置的位移，负数表示坐标值减小，正数表示坐标值增加。 即：新 x＝老 x ＋ stpx 新 y＝老 y ＋ stpy

stpw,stph	十进制数	窗口的宽度(stpw)及高度(stph)相对于上次窗口宽度及高度的变化。负数表示减小,正数表示增加。 即:新 w=老 w + stpw 新 h=老 h + stph
count	十进制数	此值定义了窗口大小及位置变化的次数。
delay	十进制数	此值定义了两次窗口的变化之间的间隔时间。 间隔=(delay)×1/60 秒

21. TRANSCOLOR

语法

TRANSCOLOR color

此命令定义了色键的颜色值,关于色键的功能请参阅后面的解释

参数	类型	描述
color	十六进制(1Byte)	色键值

22. UNFREEZE

语法

UNFREEZE

此命令使视频窗口显示活动的视频图像,即外部的视频信号开始对视频卡的显示存储区进行数据更新。

23. VCLEAR

语法

VCLEAR data

此命令将视频显示存储区用所给出的参数添充

参数	类型	描述
data	两字节	要写入视频显示区的数据值。

24. VIDEOON

语法

VIDEOON

此命令使视频显示的窗口显示功能(窗口键)和色键功能允许,从而外部输入的视频图像就可以显示在 VGA 屏幕上,关于窗口键及色键的功能请参阅后面的解释。

25. VIDEOOFF

语法

VIDEOOFF

此命令使视频显示的窗口显示功能(窗口键)和色键功能关闭,从而外部输入的视频图像就不能再示在 VGA 屏幕上显示,关于窗口键及色键的功能请参阅后面的解释。

26. VLDMMP

语法

VLDMMP filename x1 y1

此命令将一个具有"MMP"格式的文件调入 VGA 显示存储区的矩形区域内. MMP 文件格式的说明请参阅有关文件格式一章的说明。

参数	类型	描述
filename	ASCII	要调入 VGA 显示区的 MMP 数据文件名。
x1,y1	十进制数	视频卡中矩形区的左上角坐标，在范围 640×480 之内。

27. WAIT

语法

WAIT delay

此命令暂停命令解释器一段时间，然后再重新启动。

参数	类型	描述
delay	十进制数	要暂停解释的单位时间值，具体时间＝DELAY/60 秒

28. INPRECT

语法

INPRECT x1 y1 wid hgt

此命令定义在整个外部信号的 720×480 分辨率范围内的一个矩形区，采集并将其存储于视频显示区内，这个区是相对于外部视频信号而言，与视频卡的活动图像窗口没有任何关系。

参数	类型	描述
x1，y1	十进制数	外部视频信号在 720×480 空间内的矩形区左上角坐标。
wid,hgt	十进制数	将被采集并存储于视频图像显示区的外部视频信号矩形区的宽度(wid)和高度(hgt)。

29. EXTPL

语法

EXTPAL color red green blue

此命令用于定义颜色参数的值所代表的彩色内容，也就是可用此命令来修改 VGA 调色板的数据，VGA 模式 12 每一颜色值定义如下：

COLOR VALUE(十进制)	color
0	黑
1	蓝
2	绿
3	青
4	红
5	绛红
6	棕
7	浅灰
8	深灰
9	浅蓝
10	浅绿
11	淡青
12	淡红
13	淡绛红
14	黄
15	白

参数	类型	描述
color	十进制数	颜色参数值 0,1,....15。
red	十六进制(字节)	红色内容的值仅仅最低 6 位数有效,高二位必须为零。
green	十六进制(字节)	绿色内容的值仅仅最低 6 位数有效,高二位必须为零。
blue	十六进制(字节)	蓝色内容的值仅仅最低 6 位数有效,高二位必须为零。

为了应用更方便,可以编如下批量文件 DEMO. BAT

```
pcv_dem demo. scr
```

这里 DEMO. SCR 就是显示用的便笺文件。下面是它的一个例子:

```
REM 初始化
INIT
REM 定义颜色背景为白色
DEFBGND 15
REM 转移标记
LABEL LABEL1
REM 第一页  繁星点点
REM 清屏
NEWPAGE
REM
VCLEAR 42
EXTPAL 0 3f 3f 20
EXTPAL 38 3f 3f 3f
EXTPAL 3c 3f 0 0
EXTPAL 3f 0 0 20
RANDOMSTARS 25
TRANSCOLOR 3B
FITVIDEO 0 0 640 480
VCLEAR 42
SHOWVIDEO 0 0 640 480 0 0 0 0 1 0
VIDEOON
REM
UNFREEZE
GSTAR5 20 320 240 11 16 28 1 15
REM 等待 600 秒
WAIT 600

REM 第二页  开显示窗
NEWPAGE
TRANSCOLOR 3f
REM 装入 VMP 格式图像
```

```
GLDVMP logo.vmp 64 30
GLDVMP str01.vmp 640 55
REM 动画
GANIMATE 640 55 320 25 -8 0 47 1 15
GLDVMP str02.vmp 64 240
GBOX 336 248 296 226 11 15
GBOX 344 256 280 210 15 15
FITVIDEO 344 256 280 210
VCLEAR 42
SHOWVIDEO 344 466 280 0 0 -2 0 2 106 0
WAIT 600

REM 第三页  固定窗
NEWPAGE
FITVIDEO 344 256 280 210
VCLEAR 42
SHOWVIDEO 344 256 280 210 0 0 0 0 1 0
GLDVMP logo.vmp 64 30
GLDVMP str03.vmp 264 55
GLDVMP str04.vmp 64 240
WAIT 600

REM 第四页  演示压缩
NEWPAGE
VCLEAR 42
GLDVMP str05.vmp 320 55
GLDVMP logo.vmp 64 30
WAIT 100
REM
FITNSHOW 0 0 640 480 12 8 -12 -8 25 5
FITNSHOW 300 200 340 240 0 0 -12 -8 25 10
WAIT 150
FITNSHOW 0 0 640 480 0 0 0 0 1 0
WAIT 20
FITNSHOW 0 0 640 480 4 0 -12 0 40 5
WAIT 60
FITNSHOW 0 0 640 480 0 0 0 0 1 0
WAIT 20
FITNSHOW 0 0 640 480 0 6 0 -10 40 5
```

```
WAIT 600

REM 第五页　演示放不同位置
NEWPAGE
VCLEAR 42
GLDVMP logo.vmp 384 30
GLDVMP str06.vmp 16 16
FITNSHOW 360 270 240 180 －8 －2 0 0 40 5
WAIT 600

REM 第六页　演示平移
NEWPAGE
GLDVMP str03.vmp 244 60
GLDVMP str07.vmp 56 420
FITVIDEO 0 0 640 480
VCLEAR 42
REM 漫游
SHOWVIDEO 160 120 320 240 －2 0 0 0 80 0
SHOWVIDEO 0 120 320 240 0 －2 0 0 60 0
SHOWVIDEO 0 0 320 240 2 1 0 0 160 0
WAIT 600

REM 第七页　演示椭圆形窗口
NEWPAGE
GLDVMP str03.vmp 244 60
GLDVMP str08.vmp 120 420
TRANSCOLOR 3B
FITVIDEO 0 0 640 480
VCLEAR 42
SHOWVIDEO 0 0 640 480 0 0 0 0 1 0
REM 画一批圆
GCIRCLE 20 320 240 11 4 55 0 5
WAIT 600

REM 第八页　演示多次冻结框
NEWPAGE
TRANSCOLOR FF
GLDVMP str09.vmp 48 360
VCLEAR 42
```

```
SHOWVIDEO 0 0 640 480 0 0 0 0 1 0
REM 上左 1/4
FITVIDEO 0 0 320 240
TRANSCOLOR 3F
UNFREEZE
WAIT 60
REM 上右 1/4
FITVIDEO 320 0 320 240
UNFREEZE
WAIT 60
FITVIDEO 0 240 320 240
UNFREEZE
WAIT 60
FITVIDEO 320 240 320 240
UNFREEZE
WAIT 60
FITVIDEO 160 120 320 240
UNFREEZE
WAIT 600

LABEL LABELX
TRANSCOLOR 3F
REM 第九页  演示其它来源图像
NEWPAGE
FITVIDEO 320 0 320 240
VCLEAR 42
SHOWVIDEO 0 0 640 480 0 0 0 0 1 0
GLDVMP str10.vmp 56 360
VLDMMP 300E2.mmp 0 120
UNFREEZE
WAIT 600

REM 第十页  演示板
NEWPAGE
VCLEAR 42
SHOWVIDEO 160 120 320 240 0 0 0 0 1 0
GLDVMP str03.vmp 244 60
GLDVMP str11.vmp 72 400
VLDMMP vboard.mmp 160 120
```

```
WAIT 600

REM 第十一页  结论
NEWPAGE
FITVIDEO 320 240 320 240
VCLEAR 42
SHOWVIDEO 0 0 640 480 0 0 0 0 1 0
GLDVMP str03.vmp 336 32
GLDVMP str13.vmp 336 68
GLDVMP str14.vmp 336 104
GLDVMP logo.vmp 64 300
GLDVMP str12.vmp 64 390
GBOX 32 0 248 248 12 15
GBOX 36 4 240 240 15 15
VLDMMP spcman2.mmp 36 4
UNFREEZE
WAIT 600

UPDATE
JUMP LABEL1
QUIT
```

可以看出，最后一种方法给用户较多的自由。由于我们没有得到 PCV_DEM 的源程序。因此我们先分析 GRAND VIDEO 卡的硬件然后分析 PCVIDEO DOS 库编程方法，最后分析 EXAMPLE.C。在这基础上，不难编出比 PCV_DEM 更实用的软件。

第二章 一些预备知识

当读者对Grand video 卡工作性能有所感性认识后，请读者考虑一个问题，即如何把图像信号送到计算机中，并再到显示屏上显示？对于静止图像，这个问题比较简单：如果不追求速度，则利用一个硅光电池、二个步进马达、一个中速 AD 采集板和一些光源、红绿蓝三色滤光片及附属设备，就可以达到目的. 当然最先进的方案就是彩色扫描仪所用的 CCD 电子描扫方案。这两种方案的特点都是：把一 矩形图像逐行扫描成 R(tn),G(tn),B(tn)三种红绿蓝信号数字量，存入计算机内存或按一定格式存到磁盘中。反过来，再按一定规律送到显存就可以显示复制图像。这里 tn 代表不同扫描时刻，它与所测像点位置 Xn,Yn 有如下关系(用BASIC 语言表示)：

Xn＝mod(n,m)

Yn＝int(n/m)

这就是常说的逐行扫描。由于图像静止，色彩或灰度分辨率可以作得很高，画面也可以很大。对于活动图像，我们必需每秒输入 25 帧以上才没有明显的闪烁效应。早先，由于技术上的困难，于是发明了隔行扫描的技术，即奇数场只扫描奇数行，偶数场只是扫描偶数行。两场合并成一帧。它可以达到一定的分辨率又不产生抖晃感觉，信息量(通频带)又不过大。随着电子技术进步，目前不但又逐渐恢复了逐行扫描，每帧行数及每行点数也逐渐增加。比较普及型的 TV-GA 显示适配器都能达到 1024×768 256 色，高档的每帧可达到 4096×3072 65536 色。

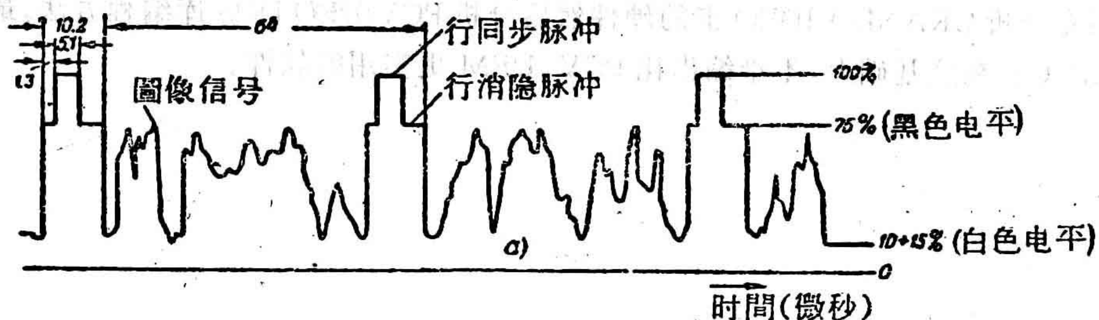


图 2.1 PAL 制复合彩色视频信号

以下我们考虑视频图像来源的放像机，摄像机或 AV，例如是单蕊电缆传送的彩色电视复合视频信号，参考图 2.1，它表示 PAL－D 制的复合电视波形图。在每行图像信号之间另有行消隐和行同步脉冲，每帧之间还有帧消隐脉冲和帧同步脉冲。这里的图像信号，以 NTSC 制式为例：

$$Em = Ey(t) + Eq\ Sin(wt + 33°)$$

$$Ey = 0.3Er + 0.59Eg + 0.11Eb$$

$$Ei = 0.6Er - 0.28Eg - 0.32Eb$$

其中 Ey 是灰度信号，Ei，Eq 分别是色差信号。

$$Eq = 0.21Er - 0.52Eg + 0.31Eb$$

Er,Eg 和 Eb 分别表示红绿蓝三色分量。

PAL 制基本上相当于对 NTSC 的色差记号，对 R－Y 记号进行了逐行倒相，以降低 NTSC 制固有的微分相位失真。

对于计算机本身的显示信号，例如目前的 VGA 或各种 Super VGA 卡都提供了 RGB 红绿蓝三种模拟色。由于人眼对色彩的分辨不如对灰度的分辨，因此电视图像信号作为载波来调幅一定频率高频信号，同时用声音对载频调频发送出去。这就是我们日常所说的电视电磁波信号：

$$V(t) = Em(t)Sin[(Wn + S(t))t]$$

$$Ei = 0.74(Er - Ey) - 0.27(Eb - Ey)$$

$$Eq = 0.48(Er - Ey) + 0.41(Eb - Ey)$$

这里 S(t)是伴声信号。

再配上行消隐和同步，就成了完全的彩电发射和接收信号。如果要把它所有细节都画出来，差不多需要一长卷，像“清明上河图”那样。图 2.2 表示由奇数到偶数行过渡的波形图的某些特征，图 2.3 表示偶数到奇数行过渡的波形图的某些特征。

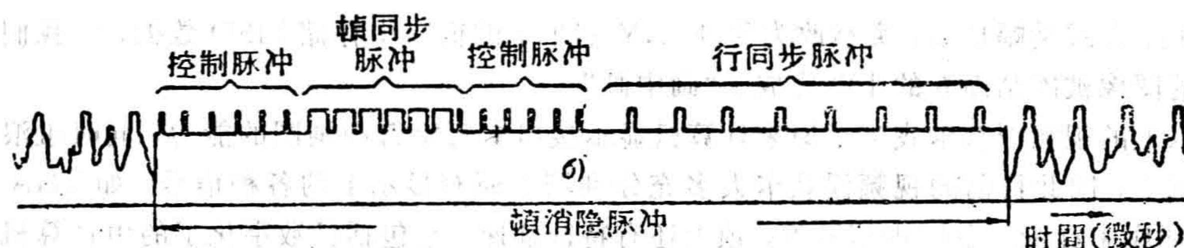


图 2.2　由奇数到偶数行过渡的 PAL 制复合彩色视频信号

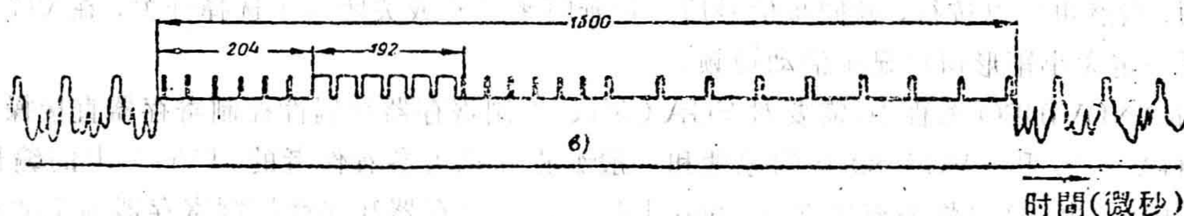


图 2.3　由偶数到奇数行过渡的 PAL 制复合彩色视频信号

为了实时显示图像及数字存储此图像(仅限一幅)，我们需要图 2.4 所示框图的数字电路硬件。

RAM 可用来存储一幅数字图像，如果图像每行选为 1024 点，共 512 行，每点 Y 用 8 位，U,V 各占 2 位，则共需内存 768K 字节，奇数偶数帧共需 0.75MB。

FIFO 用来快速存储每行数字图像信息，在每行回扫期间，它的信息可以转存到图像 RAM 中。当存储每帧信号到视频 RAM 中，适时地把它的数据取出来经 LUT 便查表用来对

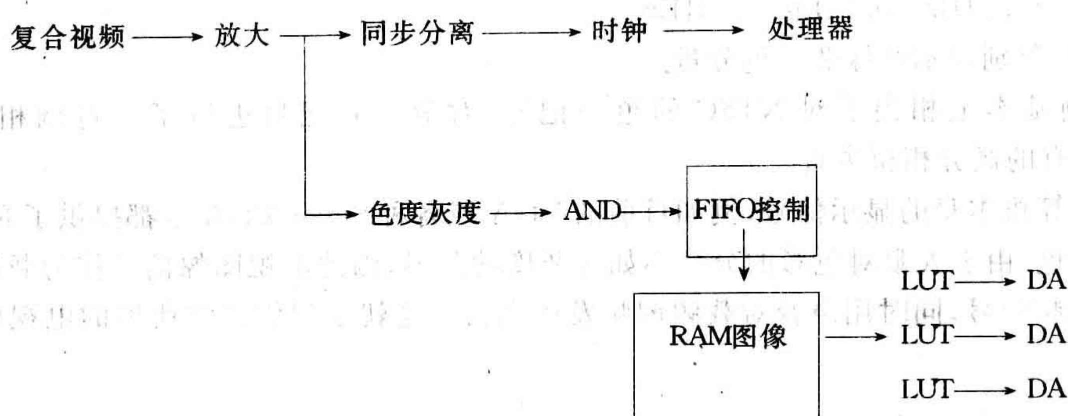


图2.4 视频俘获原理图

红、绿、兰三色进行变换后，送去DA转换成模拟信号分别显示各自的色彩。

显然，显示时间比输入活动视频信号图像要慢些，最多落后为扫描一帧的时间，延迟最小时间为扫描一行的时间。

如果要把活动视频图像在显示屏上从(0,0)平移到(X,Y)位置，可以在FIFO向RAM传送时分别增加一行(Y)列(X)地址来达到。当图像缩小比例为1/2，1/3，1/4.....时还可以在FIFO向视频RAM转储数据时每隔i/n个点或每隔n扫描行存储一次FIFO来实行。如果原来以1:1方式满幅显示，突然改为移到X,Y缩小了的窗口中存储FIFO数据，则我们观察到原先的图像被冻结而新的小图像成了"画中画"。

以上的视频俘获卡丧失了原来计算机显示接口卡写字符和画图的能力，同时也浪费了一块显示卡，因此目前的视频俘获卡大多充分利用了原有显示卡的各种信号，如VGA卡除了15针插座中红绿兰及同步信号外，顶上还有特征插座，它包括已数字化了的由计算机程序产生的视频图形数据8位。

由于计算机产生图形视频信号与活动视频信号不一定同步，当VGA信号与活动视频信号混合时，需要改变图3中输出的时序(Skey)使它与VGA的信号同步。

另外需加一个高速模拟开关，当VGA中视频数据等于某色X值时，分别选择活动视频或VGA图形。用这种方法可以VGA对应图像在显示屏中开任意形状窗口，在窗口中显示活动视频。当然也可以按行，帧同步信号后一定延时来打开或关闭这个选择开关，在VGA显示窗中开一定大小矩形窗口显示活动视频。

GRANDVIDEO编程中，需要对VGA CRTC控制寄存器和属性控制寄存器直接操作。关于TVGA 9000和TVGA 8900的原理和一般编程知识可参看作者的《TVGA卡的编程应用开发入门》一书。这里将补充TVGA 8900卡的CRTC寄存器和属性控制寄存器为主的有关知识。TVGA的CRTC寄存器和VGA的CRTC基本相同，而后者已有不少专书介绍，我门在这里只介绍与TVGA 8900显示视频水平和垂直时序有关的一些应用例子。对于GRAND VIDEO的应用编程，目前只用到CRTC的水平和垂直回扫及属性寄存器等信息。我们有意地介绍一些较为专门的知识。目前的GRANDVIDEO卡编程还不需要，但是如果想设计一种1024×1024或1280×1024显示方式的高分辨视频俘获二合一卡，这里的数据可能会有参考价值。

表2.1表示对几种显示器TVGA CRTC的参数设定值及CRTC各寄存器的参数功能。

* *

显示模式	类型	列数	列数	分辨率	颜色	隔行逐行	混合输出	模式控制寄存器 2		
							3C2	3C5.0D	3C5.D0	3C5.D
日立 HITACHI Super Scan 显示器 TVGA CRTC 寄存器参数										
5e	Graph	100	37	800×600	256	NO	11100011		00000000	01000001
5f	Graph	128	48	1024×768	16	NO	00100111		00000000	00000001
62	Graph	128	48	1024×768	256	NO	00100111		00010000	00000001
日立 HITACHI Super Scan 带 70 赫兹板 TVGA CRTC 寄存器参数										
							11100011		同上	同上
							00100111			01000001
							00100111			1
三菱 MITSUBISHI Diamond Scan TVGA CRTC 寄存器参数										
							同上		同上	01000001
										00000001
										00000001
三菱 MITSUBISHI Diamond Scan 带 70 赫兹板 TVGA CRTC 寄存器参数										
							同上		同上	同 1
Any—Maker 1024x768 逐行扫描板(70Hz) TVGA CRTC 寄存器参数										
							同上		同上	同 1
Any—Maker 1024x768 隔行扫描板(60Hz) TVGA CRTC 寄存器参数										
3							01100011			
12							11100011			
5e							同上		同上	同 0
5f										
62										
Any—Maker 1024x768 隔行扫描板 TVGA CRTC 寄存器参数										
5e							11100011		同上	01000001
5f							00101011			0
61				768x1024 I			00101011			
62							00101011			0

3C2 :位 0—1CRTC 地址为 3DXH,彩色仿真;

位 1—1 允许 RAM 存取;

位 3,2—00:25.175MHZ,640 像素;

01 28.322 720

10:外时钟(最大 65MHZ) 11:外时钟(最大 65)

位 5—1:奇偶模式(1—5)下在两个 64K 内存页之间选高地址页;

位 6—0:正水平回扫;1:负回扫
位 7—0: 正垂直回扫;1:负回扫

3C5.0D:新模式控制寄存器:
位 0—0 选择 3C2 中的时钟 ;1 选择新时钟
位 2—1:输入时钟分频系数 00:除 1 01:除 2 10: 除 4 11: 除 1.5
位 3—BIOS 操作时钟分频 0:除 1; 1:除 2
位 5,4—BIOS

* *

		电源开关 模式 2	测试寄存器	软件可 编程	水平总计
3C5.E0	3C5.E1	3C5.0F	3D5.1E	3D5.1F	3D5/3B5 0
TVGACRTC0 HITACHI Super Scan					
10110100	01000000	11111000	10000000	00011001	11111011
10101000	01000000	01110100	10000000	00011001	10100010
10100000	0	11110100	10000000	00011010	10100010
TVGACRTC 1 HITACHI Super Scan with 70Hz Board					
10110000	0	11111100	同 0	00011001	11111011
10111000	0	11110000		00011011	10100100
10111000	0	11110000		00011011	10100100
TVGACRTC 2 MITSUBISHI Diamond Scan					
同 0	同 0	同 0	同 0	同 0	同 0
TVGACRTC 3 MITSUBISHI Diamond Scan with 70Hz Board					
同 1	同 1	11111100	同 0	同 0	同 1
		11111100			
		11110000			
TVGACRTC 4 Any—Maker 1024x768 NonInterlaced(70Hz)					
同 1	同 1	同 1	同 0	同 1	同 1
TVGACRTC 5 Any—Maker 1024x768 NonInterlaced(60Hz)					
					01011101
					01011111
同 0	0	11111100	同 0	同 0	同 0
	01000000	01110100			
	0	11110100			
TVGACRTC 6 Any—Maker 1024x768 Interlaced					
10110000	同 1	11111100	10000000	同 0	11111011
10100000		11111100	10000100		10011001
					10001111

10100000　　11110100　10000100　　10011001

3C5.0F:电源开关模式寄存器 2

位 0—3:存储卡上 DIP 开关信息:0—ON ;1—OFF

位 4　:0 选微通道总线;1 PC/AT 总线

位 5　:1 正常地址映射

位 7　:0 8 位 BIOS;1 16 位 BIOS

3D5.1E:CRTC 模块测试寄存器

位 2:0 选择非交替模式;1 选择交替模式

位 7:1 允许 16 位地址

3D5.1F:软件可编程寄存器

位 01 :TVGA 上显存容量 00:256K 11:1024K

位 3:

3D5.0:水平字符总计(最小为 5)

* *

水平显示结束	水平消隐开始	水平消隐结束	水平回扫开始	水平回扫结束	垂直总计	溢出	预置行计数
3D5.1	2	3	4	5	6	7	8
TVGACRTC　0 HITACHI　Super Scan							
199	201	205	209	131	113+512	11110000	0
127	128	133	135	144	44+512	11111101	0
127	128	133	135	144	44+512	11111101	0
TVGACRTC　1 HITACHI　Super Scan with 70Hz Board							
		10011101	11001111	同	113+512	同	
全同	同	10000111	10000100		37+512		
		10000111	10000101		37+512		
TVGACRTC　2 MITSUBISHI Diamond Scan							
	同	同 0	同 0	同	同 0	同	
TVGACRTC　3 MITSUBISHI Diamond Scan with 70Hz Board							
	同	同 1	11001111	同	113+512	同	
			10000011		33+512		
			10000011		33+512		
TVGACRTC　4 Any—Maker 1024x768 NonInterlaced(70Hz)							
	同	同 1	同 1	同	同 1	同	

TVGACRTC 5 Any—Maker 1024x768 NonInterlaced(60Hz)						
01001111						
01001111						
	同	同 0	207	同	同 0	同
			167			
			167			
TVGACRTC 6 Any—Maker 1024x768 Interlaced						
	201	157	207	163	113+512	11110000
	161	27	163	16	157+256	00011111
95	97	27	97	28	28+512	10110010
	161	27	163	25	152+256	00011111

3D5/3B5(CRTC)

3D5.1:水平显示允许结束寄存器(字符数)+

3D5.2:水平空白开始寄存器(字符数)

3D5.3:水平空白结束寄存器

3D5.4:水平回扫开始寄存器(字符数)

3D5.5:水平回扫结束寄存器

3D5.6:垂直总计像素数减二(十位中低八位)

3D5.7:0 垂直总计的第八位，5 第九位

1 垂直显示结束八位，6 第九位

2 垂直回扫开始八位，7 第九位

3 垂直消隐开始八位，

4 行比较的第八位

3D5.8:预置行扫描寄存器:0—4 预置行扫描

5,6 水平平移

* *

字符扫描线数	光标开始字符行	光标结束字符行	显存开始地址高	低	光标位置高	低	垂直回扫开始低
3D5.9	A	B	C	D	E	F	10
TVGACRTC 0 HITACHI Super Scan							
01100000	0	0	0	0	0	0	90+512
01100000	0	0	0	0	0	0	15+512
01100000	00011101	01111111	0	0	0	0	15+512
TVGACRTC 1 HITACHI Super Scan with 70Hz Board							
同	00011101	01111111					90+512
	00011101	01111111					7+512
	00011101	01111111					7+512
TVGACRTC 2 MITSUBISHI Diamond Scan							
同	同 0	同 0					同 0

TVGACRTC 3 MITSUBISHI Diamond Scan with 70Hz Board

同	同 1	同 1					90+512
							3+512
							4+512

TVGACRTC 4 Any—Maker 1024x768 NonInterlaced(70Hz)

同	同 1	同 1					同 1

TVGACRTC 5 Any—Maker 1024x768 NonInterlaced(60Hz)

同	00011101	01111111					同 2
	0	0					
	00011101	01111111					

TVGACRTC 6 Any—Maker 1024x768 Interlaced

01100000	00011101	01111111					90+512
0	00011101	01111111					132+256
01100000	0	0	0	0	0	0	4+256
0	00011101	01111111					129+256

3D5.9:每字符最大扫描线数 0—4
5 垂直消隐开始的第九位
6 行比较的第九位
7200 400 线转换

3D5.A:光标开始字符行 0—4
5:0 开光标 1 关光标

3D5.B:光标结束字符行 0—4
5,6:光标信号对于光标位置的偏移时钟数

3D5.C:显存开始地址高位
3D5.D: 低位
3D5.E:光标位置高位
3D5.F: 低位
3D5.10:垂直回扫开始低八位

* *

垂直回扫结束高	垂直显示结束	偏移/罗辑屏宽	下划线定位	垂直消隐开始	垂直消隐结束	模式控制	行比较计数器
3D5.11	12	13	14	15	16	17	18

TVGACRTC 0 HITACHI Super Scan

10001100	01010111	01100100	01000000	01011100	01101100	10100011	11111111
10000001	11111111	01000000	0	00000111	00100110	11100011	11111111
10000001	11111111	01000000	01000000	00000111	00100110	10100011	11111111
TVGACRTC 1 HITACHI Super Scan with 70Hz Board							
10001100	同	同	全同	同	01101100	全同	全同
1					00011111		
1					00011111		
TVGACRTC 2 MITSUBISHI Diamond Scan							
同 0	同	同		同	同 0		
TVGACRTC 3 MITSUBISHI Diamond Scan with 70Hz Board							
同 1	同	同		同	01101100		
					00011011		
					00011011		
TVGACRTC 4 Any－Maker 1024x768 NonInterlaced(70Hz)							
同 1	同	同		同	同 1		
TVGACRTC 5 Any－Maker 1024x768 NonInterlaced(60Hz)							
同 0	同	同		同	同 0		
TVGACRTC 6 Any－Maker 1024x768 Interlaced							
10001100	01010111	01100100		01011100	01101100		
10000110	01111111	10000000		10000100	10011000		
138＋384	11111111	01100000	0	4	24 * 256	227	11111111
10000110	01111111	10000000		10000011	10010101		

3D5.11:垂直回扫结束 0－3 位

4:0 清垂直中断

5:0 允许垂直中断

6:0 每水平线执行 5 个 DRAM 刷新

7:0 允许写 CRT 寄存器 0－7 位

3D5.12:垂直显示允许结束的低八位

3D5.13:偏移/逻辑屏幕宽度

3D5.14:下划线定位 0－4

5 隔 4 计数

6:1 双字模式

3D5.15:开始垂直消隐

3D5.16:结束垂直消隐

3D5.17:模式控制寄存器

0:支持兼容模式

1:选择行扫描计数器0 位 1 是MA14

1　　　MA14 是

2:水平回扫选择:1 水平回扫除以二

3:隔二计数:1 存储地址计数用字符时钟除以二

5:地址环绕,内存地址计数器 13 或 15 位送到 CRT 控制寄存器的 MA0 线

6:下划线第六位为零时 0:选字节方式 1 字方式

3D5.18:行比较计数器的低八位,用来分屏为二.

表 2.2　表示 TVGA 图形控制寄存器在不同显示模式下参数设定值

* *

图形控制寄存器　索引 3CE　数据 3CF

模式	设置重置	允许 RS	色彩比较	操作环移	读映射选择	模式	混合	忽略色彩比较
	0	1	2	3	4	5	6	7
3	0	0	0	0	0	00010000	00001110	00000000
12	0	0	0	0	0	00000000	00001001	00001111
5B	0	0	0	0	0	01000000	00000101	00001111
5C	0	0	0	0	0	01000000	00000101	00001111
5D						同上		
5E								
5F						00000000	00000101	00001111
60		同上				00000000	00000101	00001111
61						00000000	00000101	00001111
62						01000000	00000101	00001111

* *

定序寄存器　3C4　3C5

模式	0	1	2	3	4
3	00000011	00000001	00000011	00000000	00000010
12	00000011	00000001	00001111	00000000	00000110
5B	00000011	00000001	00001111	00000000	00000110
5C	00000011	00000001	00001111	00000000	00001110
5D	同上				
5E					
5F	00000011	00000001	00001111	00000000	00000110
60	00000011	00001001	00001111	00000000	00000110
61	00000011	00000001	00001111	00000000	00000110
62	00000011	00000001	00001111	00000000	00001110

表 2.3 表示 TVGA 属性寄存器的参数设定值

* *

属性控制寄存器 3C0 3C1

模式	0	1	2	3	4	5	6	7	8	9	A
3	0	1	2	3	4	5	00010100	7	00111000	9	00111010
12	同上										
5B											
5C							6	7	8	9	A
5D							同上				
5E											
5F							同 3				
60					00010111	5	00010100	7	00111000	9	00111010
61	同 3										
62	同 5C										

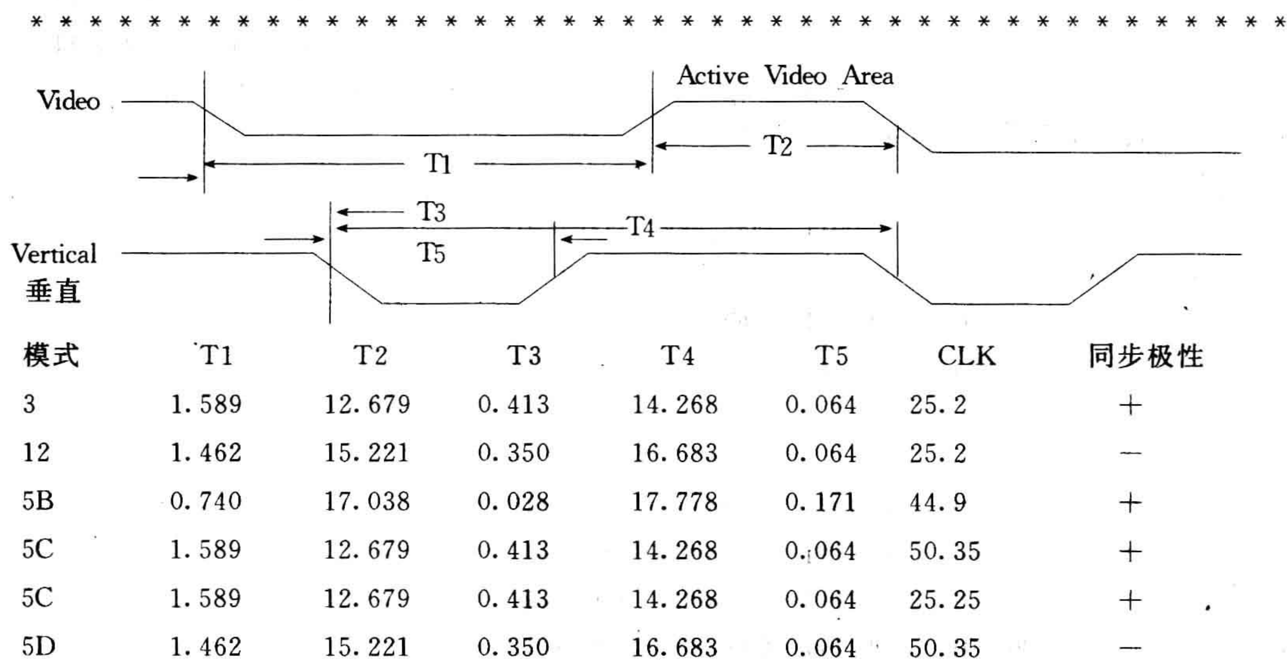
* *

属性控制寄存器 3C0 3C1

模式	B	C	D	E	F	10	11	12	13	14
3	B	3C	D	3E	F	C	11	F	13	0
12	B	3C	D	3E	F	1	11	F	13	0
5B	B	3C	D	3E	F	1	11	F	13	0
5C	B	C	D	3E	F	41	11	F	13	0
5D							同上			
5E										
5F							同 12			
60	B	3C	D	3E	F	1	11	5	13	0
61	同 12									
62	B	C	D	E	F	41	11	F	13	0

表 2.4 是 TVGA 的垂直时序

* *

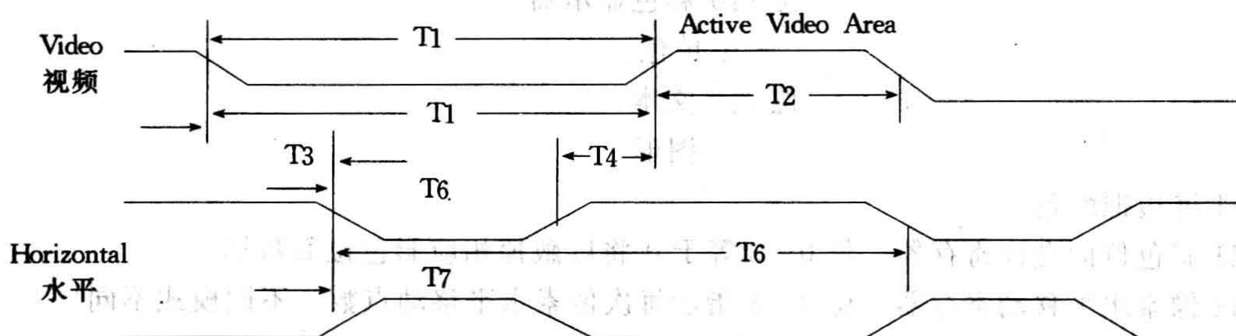


模式	T1	T2	T3	T4	T5	CLK	同步极性
3	1.589	12.679	0.413	14.268	0.064	25.2	+
12	1.462	15.221	0.350	16.683	0.064	25.2	−
5B	0.740	17.038	0.028	17.778	0.171	44.9	+
5C	1.589	12.679	0.413	14.268	0.064	50.35	+
5C	1.589	12.679	0.413	14.268	0.064	25.25	+
5D	1.462	15.221	0.350	16.683	0.064	50.35	−

5D						25.25	
5E	0.999	19.105	0.121	20.104	0.091	65	−
5E	0.740	17.038	0.028	17.778	0.171	36	−
5F	0.901	22.465	0.141	23.366	0.225	44.9	+
5F	1.783	15.198	0.476	16.982	0.099	65	+
60	0.958	23.887	0.15	24.845	0.239	44.9	+
61	0.817	27.767	0.132	28.585	0.158	44.9	+
62	0.901	22.465	0.141	23.366	0.225	44.9	+
62	0.921	16.048	0.230	16.969	0.335	65	+

表 2.5 TVGA 的水平时序

* *



模式	T1	T2	T3	T4	T5	CLK	同步极性
3	6.778	25.104	1.907	0.953	3.813	25.2	−
12	6.673	25.104	1.589	1.271	3.813	25.2	−
5B	6.444	22.000	1.333	3.378	1.333	44.9	+
5C	6.514	25.263	1.112	1.271	4.131	50.35	+
5C	6.673	25.104	1.589	1.271	3.813	25.25	+
5D	6.514	25.263	1.112	1.271	4.131	50.35	−
5D	6.673	25.104	1.589	1.271	3.813	25.25	
5E	5.785	24.492	0.492	1.969	3.323	65	−
5E	6.444	22.000	1.333	3.778	1.333	36	−
5F	5.523	22.628	0.713	2.494	2.316	44.9	+
5F	4.185	15.631	0.492	2.092	1.600	65	+
60	3.742	26.192	0.891	1.425	1.425	44.9	+
61	9.443	16.927	0.356	4.276	4.811	44.9	+
62	5.523	22.628	0.713	2.494	2.316	44.9	+
62	5.292	15.631	0.985	1.846	2.462	65	+

2.1 TVGA 属性控制器

属性控制寄存器的输出接到 TVGA 视频 DAC 的输入，产生红绿蓝信号。它内部共有 20 个寄存器，对应索引地址 0 到 14H。属性控制寄存器的读写口地址都是 3C0H，它的第一次读写是指向索引地址，以后是该索引地址对应的寄存器的数据。每读写 3C0 口一次，交替在索引，数据之间翻转。属性控制寄存器的清除可由向 3DA(彩色方式)口读写来实现。属性控制寄存

器的索引寄存器只用了低五位作索引地址，D5 是调色板源：0 表明修改调色板，清屏；1 保持调色板不变允许显示。索引地址 0 到 0F 对应于 0 到 15 号彩色调色板寄存器。以下索引地址对应的寄存器如下表所示：

索引地址	寄存器名	功能
10H	模式控制寄存器	位 7：0 视频 DAC 输出 P4 P5 来源是调色板寄存器 1 颜色选择寄存器的 D0 和 D1 位 6：像素宽度设为 1 位 5：1 水平移动可以 位 3：0 底色置成高亮度 1 闪烁属性 由字符属性第 7 位赋能 位 1：0 彩色显示器 1 单色 位 0：0 文本 图形

11H 屏边框颜色

12H 彩色位面允许寄存器　位 0—3 等于 0 将屏蔽掉相应彩色位面数据。

13H 像素水平移动寄存器　位 0—3 指示每次像素水平移动点数。不同模式不同！

14H 颜色选择寄存器　位 3：颜色 7

位 2：　6　DAC 的高位

位 1，0：模式控制寄存器的 P5 P4 输出

2.2 I2C 总线

I2C 总线是数字电视技术中采用的一种串行总线。

2.2.1 I2C 总线说明（包括快模式）

一、引言

对于八位应用，需用单片微处理器，应建立如下设计标准：

(1) 一个完整的系统通常至少由一个微处理器及其它的外围设备如存储器及 I/O 扩展设备等组成。

(2) 系统内联结诸设备的花费最小。

(3) 这样的一个系统通常有控制功能并且不需要高速数据传送。

(4) 总效率取决于所选择的设备及相互联结的总线结构。

为了产生一个满足这样标准的系统，需要串行结构。虽然串行总线的吞吐量不如并行总线，但需要较少的联结管脚。然而，一个总线不只是联结线，它体现出在系统内所有的通讯格式与过程。为防止所有可能的混乱、数据丢失和信息受阻，设备间的通讯必须有一定形式的协议。系统必须不能依靠联结于它的设备，否则修改与改进是不可能的。一个过程也要设计好以决定哪一个将控制总线以及何时控制。如果联于总线的不同设备有不同的时钟速度，必须定义总线时钟。所有这些标准贯穿于 I2C 总线说明的始终。

二、I2C 总线概念

任何 IC 电路(NMOS，CMOS，双极性)都可由 I2C 总线支持。串行数据(SDA)与串行时钟(SCL)两条线在联于总线上的设备间传载信息。每个设备唯一的由一个地址确定，无论它是微处理器、液晶显示驱动器、存储器还是键盘接口。一个设备是发送者还是接受者取决于该设备的功能。显然，液晶显示驱动器是一个接受者，而存储器既可接受又可发送数据。除了发送器与接受器外，设备在实现数据传送时也可看作主设备或从设备。在总线上启动一个数据，并产生时钟信号以允许发送的设备为主设备，此时由地址选通的任何设备均为从设备。

三、I2C 总线术语的定义

发送器：送数据到总线的设备。

接受器：从总线接受数据的设备。

主设备：初始化一个传送，产生时钟信号及终止一个传送的设备。

从设备：被一个主设备选址的设备。

多主设备：多个设备在同一时刻可试图控制总线而不使信息紊乱。

仲裁判优：是一个这样的过程，如果多个主设备同时试图控制总线(图 2.5)，只有一个被允许而不使信息紊乱。

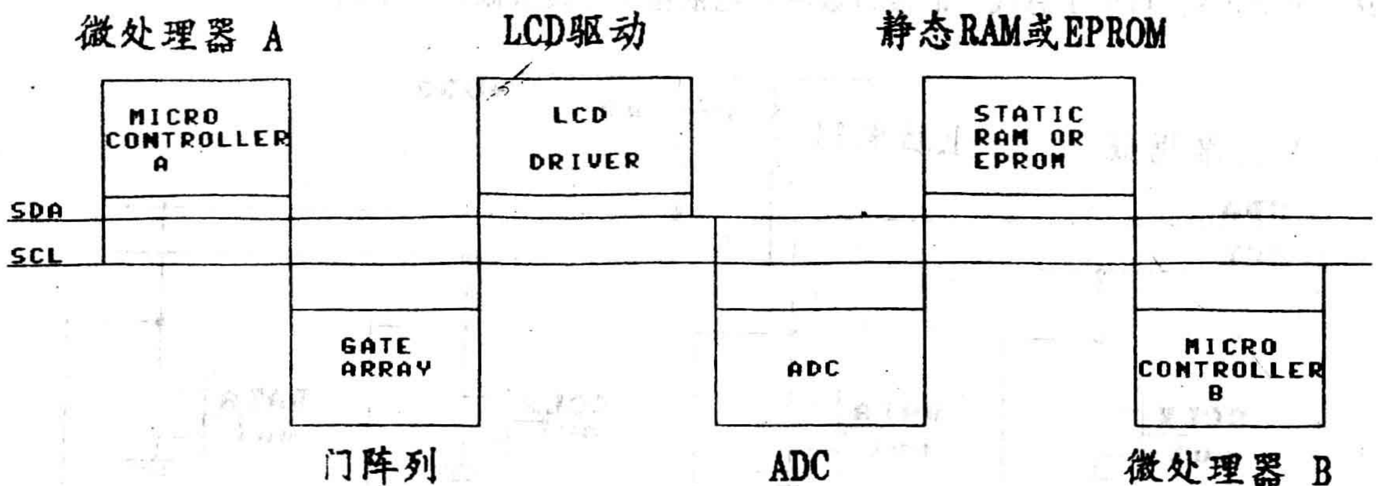


图 2.5 I2C 总线结构的例子

同步化：使两个或多个设备同步的过程。

I2C 总线为多主总线。这意味着联于总线上的主设备多于一个。由于主设备通常都是处理器，主从设备及接受器发送器在 I2C 总线上的关系仅取决于此时数据传送的方向。数据传送的过程如下：

(1) 假如微处理器 A 要送信息到微处理器 B：

微处理器 A(主)找到微处理器 B(从)的地址；

微处理器 A(主发送者)送数据到微处理器 B(从接受者)；

微处理器 A(主)终止传送。

(2) 假如微处理器 A 需要从微处理器 B 接受数据：

微处理器 A(主)找到微处理器 B(从)的地址；

微处理器 A(主接受者)接受数据从微处理器 B(从发送者)；

微处理器 A(主)终止传送。

在这种情况下，主设备(微处理器 A)产生时序并中断传送。

联于 I2C 总线上多于一个微处理器的可能性，意味着在同一时刻会有多个主设备试图启动一个数据。为避免由此引起的混乱，使用了在总线上所有 I2C 接口都用联线与的判优法过程。如果两个以上的主设备试图送信息到总线，则首先淘汰掉此时信号为高电平的主设备。在判优时，时钟信号为诸设备所产生的时钟通过联于 SCL 线上与电路的同步结合(关于判优的具体方法祥见 6.0)。

I2C 总线上时钟的产生与各主设备有关，在总线上传送数据时每个主设备产生自己的时钟。总线上一个主设备的时钟信号只有当其被一个慢的从设备的低电平延长而改变，或在判优发生时被另一个主设备代替。

2.2.2 一般特征

SDA 与 SCL 都是双向线，它们通过一个提拉负载电阻联于正电源。当总线为空闲时它们都为高。联于总线上的设备在输出级必须具有开耗尽(open－drain)或开路集电极的功能，以实现与电路的功能。在标准模式下，I2C 总线上的数据传送速率可达 100Kbit/s，在快模式下可达 400Kbit/s。可联于总线上的接口数唯一地依赖于总线限制电容 400pf。

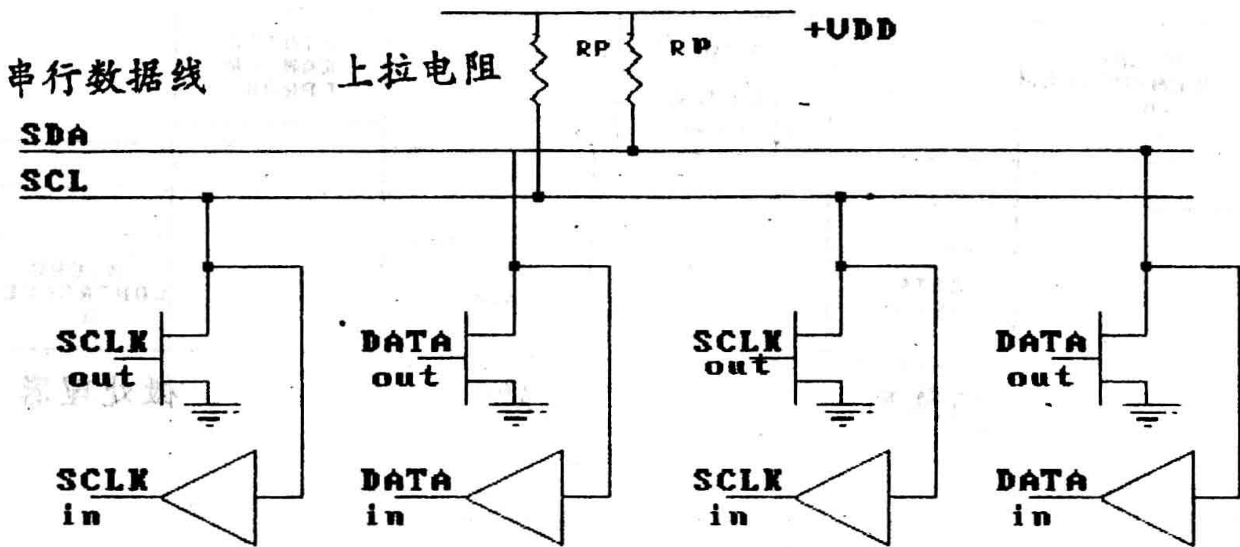


图 2.6 I2C 设备到 I2C 总线的联接

2.2.3 位传送

由于联于 I2C 总线上的元件有不同技术指标的产品(CMOS，NMOS，双极性)，电平的逻辑"0"(低)和"1"(高)并不固定，它们与 Vdd 的电平有关。传送一个数据位需要一个时钟脉冲。

2.2.4 数据的有效性

在时钟高时 SDA 线上的数据必须稳定。数据线上的高或低状态只有当时钟信号在 SCL 线上为低时才能改变(图 2.7)。

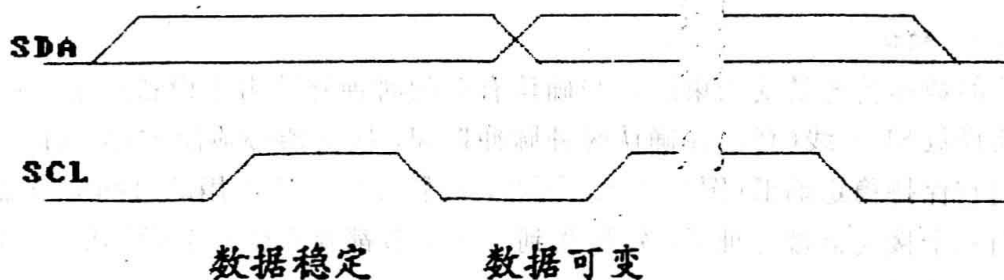


图 2.7　I2C 总线上的位传送

2.2.5　起始条件和停止条件

在 I2C 总线过程中，定义当 SCL 为高时 SDA 由高到低为起始条件，而当 SCL 为高时 SDA 由低到高为停止条件。起始和停止条件总是由主设备产生。在起始条件后总线被认为是忙的，在停止条件后总线被认为是闲的。总线闲状态将在以后的 2.7.5 中说明。

起始和停止条件很容易被联于总线的设有必需接口的设备检测到。然而，没有此接口的微处理器必须至少每个时钟周器在 SDA 线上采 2 次以检测这种变化。

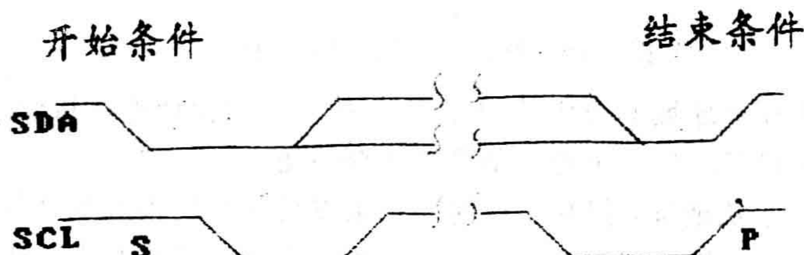


图 2.8　开始条件及结束条件

2.2.6　传送数据

2.2.6.1　字节格式

送到 SDA 线的每个字节必须 8 位长。每次传送的字节数不受限制，每个字节后要跟一个确认位。先传最高位(图 2.9)，如果接受器直到要行使别的功能如服务内部中断时仍未收到这个数据的完整字节，它将保持时钟线 SCL 低，以强制发送者进入等待状态。到了接受器处于准备好状态时，数据传送接着进行，继续接受这个数据的低位，并释放时钟线 SCL。

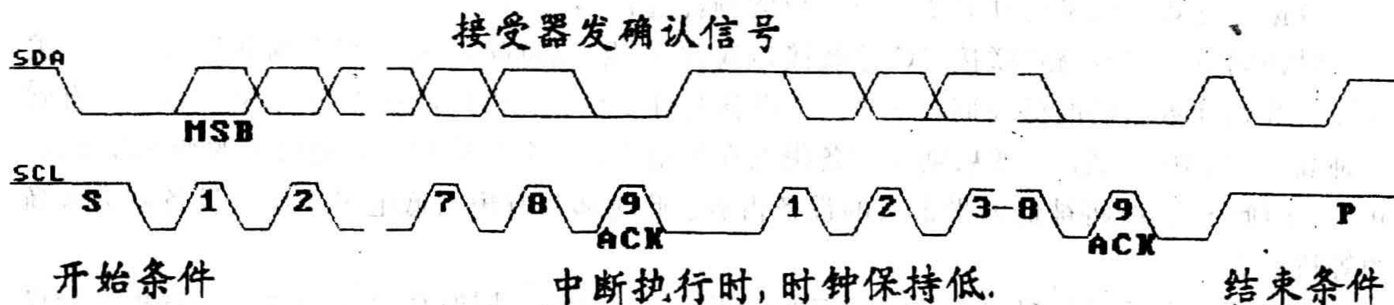


图 2.9　I2C 总线上的数据传送

在有些情况下，I2C 总线使用不同的格式(如对 CBUS 兼容的设备等)是允许的。以这样的地址开始的信息可被停止条件停止，即使在传送一个字节时也是这样。在这种情况下没有确认产生。

2.2.6.2　确认

有确认的数据传送是受约束的。与确认有关的时钟脉冲由主设备产生。在确认时钟脉冲期间发送器释放 SDA 线(高)。在确认时钟脉冲期间，接受器必须使 SDA 线低，并且在时钟脉冲为高期间它保持稳定的低(图 2.10)。当然，设置与保持所占用的时间应考虑到。

通常当一个接受器被寻址后，每接受到--个字节都要产生一个确认信号。(除非信息的开始使用 CBUS 地址。

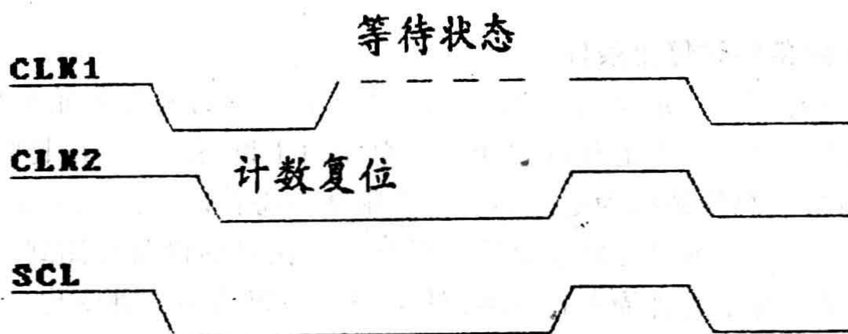


图 2.10　判优过程中时钟同步化

当一个从接受器不在从地址上确认时(如由于行使实时功能而不能接受)，数据线则被从接受器升为高。此时主设备产生一个停止条件以放弃传送。

如果从接受器确认了从地址，但晚一些时间在数据传送中还没收到任何数据，主设备必须再一次放弃传送。这由从接受器在第一字节后产生一个不确认来指明，从设备使数据线为高，主设备则产生停止条件。

如果一个主接受器卷入一个数据传送中，它必须以在最后字节后，不产生确认对从发送器产生一个结束数据的信号。从发送器必须释放数据线以同意主设备产生停止条件。

2.3　判优和时钟产生

2.3.1　同步化

所有的主设备产生它自己的时钟到 SCL 线，以便在 I2C 总线上传送信息。只有在时钟为高时数据才有效。定义时钟需要一位一位的判优过程。

时钟同步由"与电路"联接 I2C 总线接口到 SCL 线。这意味着 SCL 线上的下降沿引起相关的设备开始计数它们的低周期，只要一个设备时钟变低，它将以此状态占有 SCL 线，一直到时钟高(图 2.10)。然而，如果别的设备仍然在低电平，这个时钟的上升沿将不改变 SCL 线的状态。因此 SCL 线将被低电平最长的设备占有。此时那些有短的低电平周期的设备将进入高的等待状态。

如果所有有关设备都同时计完它们的低周期，时钟线将同时释放并为高。然后将不分设备时钟与 SCL 线时钟，所有设备将开始计数它们的高周期。第一个计完它的周期的设备又使 SCL 线变低。同步时钟就是这样产生的。其低电平由诸设备中低电平最长的决定，其高电平由诸设备中高电平最短的决定。

2.3.2 判优

一个主设备只有当总线为闲时才可进行传送。2 个或更多的主设备可在最小的 START 条件占有时间($T_{HD.STA}$)内产生 START 条件，导致 START 条件到总线。

当 SCL 线为高电平时，判优发生在 SDA 线上。这样，一个主设备在别的主设备发送低时它发送高电平，则关闭其数据输出，这是由于总线上的电平不响应它自己的电平。

判优可继续进行多位，第一个阶段是比较地址。如果主设备都选同一个设备，则判优继续在数据上比较。在 I2C 上地址与数据信息用来判优。此过程不丢失任何信息。

在判优期间主设备可产生时钟，直到被淘汰。如果一个主设备兼有从设备的功能，它在判优的选址阶段被淘汰。有可能获胜的主设备会寻址于它。被淘汰的主设备必须立刻转向从接受模式(图 2.11)。

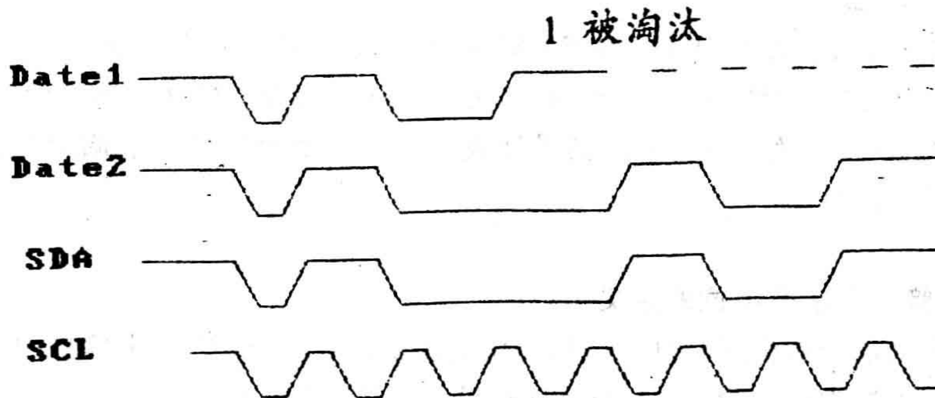


图 2.11 判优过程中的两个主设备

在串行传送时要特别注意，在重复的 START 条件或 STOP 条件向 I2C 总线发送时，判优过程仍在继续。下属情况发生是可能的：卷入的主设备必须送它的重复 START 条件在格式 FRAME 中的同一位置。换句话说，判优不允许在以下两个之间：

(1) 一个重复 START 条件与一个数据位；

(2) 一个 STOP 条件与一个数据位；

(3) 一个重复 START 条件与一个 STOP 条件。

2.3.3 使用时钟

作为握手信号的同步机能。

为了在判优中被使用，时钟同步机能可被用来使能接受器以适合快模式数据传送，无论是字节还是位。对字节，一个设备也许有能力快速接受字节，但要用较长的时间存储接受的字节或准备另外要被传送的字节。从设备在接受数据和确认后要使 SCL 线低，以强使主设备进入等待状态直到准备好下一个握手过程中的信号传送。对位，一个设备如微处理器，在其芯片上没有或仅有一个限制的硬件 I2C 接口，延长每一个时钟的低周期可使总线时钟减慢。这样，任何主设备的速度都被改变为这个设备的内部操作速率。

2.4 7位地址格式

在START条件后，送一个从地址，其前7位是地址位，第8位(R/W)表示数据方向，0表示发送(WRITE)，1表示请求(READ)。数据传送由主设备产生的STOP条件终止。如果主设备仍然要在总线上通讯，它可产生一个重复START条件(Sr)，选址另一个从设备而无需首先产生一个STOP条件。由这样的传送可组成多种读/写格式：(1) 主发送器传到从接受器,传送方向不变。(2) 在第一个字节后主设备立刻读从设备。在第一个确认时，主发送器变成接受器，而从接受器则变为发送器。这个确认仍然由从设备产生，STOP条件由主设备产生。(3) 组合的格式。

在传送改变方向时，START条件和从地址都重复，但R/W位保留。

2.4.1 7位选址

I2C总线的选址过程是这样的：在START条件后的第一字节决定哪一个从设备被主设备选中，所例外的是“一般调用”。“一般调用”可选址所有设备，当这个地址使用时，理论上所有设备都由确认响应。然而，设备可以被设计成不响应这个地址。“一般调用”的第二个字节定义必须执行的操作信息。

2.4.2 在第一个字节中位的定义

第一个字节的前7位构成从地址。第8位是最低位(LSB)，它决定信息的方向，0表示主设备将把信息写入选中的从设备，1表示从从设备中读(图2.12)。

一般调用地址　　第二字节　　字节和确认

S	00000000	A	Master Add	1	A	Date	A	Date	A	P

图2.12 从一个硬件主发送器传输数据

当一个地址被送后，系统中每一个设备在START条件后与自己的地址相比较。如果自己与该地址一样，则认为是被主设备选中的，是从接受器或从发送器取决于R/W位。

一个地址由固定的和可程序的部分组成。由于系统中有可能有几个相同的设备，从地址中可程序部分使得联于I2C总线的这样的设备数达到最大。一个设备可程序的地址位数决定于它的有效管脚数。如果一个设备有4个固定位和3个可程序的地址位，则总共可有8个这样的设备联于同一总线。

2.4.3 一般调用地址

一般调用地址用来联于I2C总线上的每一个设备。如果某个设备在一般调用时不需要任何数据，它可以用不确认来不响应这个地址。如果某个设备需要数据，则确认这个地址并象从接受器一样操作。第二个及以后的字节将被每个有能力接受这个数据的从接受器确认。一个不能处理其中之一字节的从接受器必须用不确认来不响应。一般调用的意义包含在第二个

字节里。有两种情况要考虑：(1) 第二(低)字节的位B为0；(2) 第二(低)字节的位B为1。

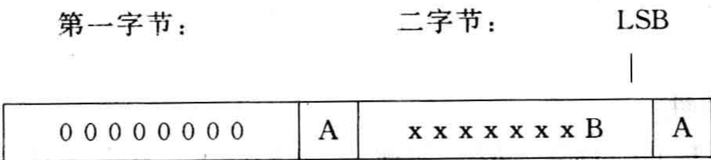


图 2.13 一般调用地址格式

当B为0时，第二字节有下列定义：

——00000110(06H) 从设备地址由硬件复位并写可程序的部分。在接收到这两个连续的字节后，所有被设计可响应一般调用的设备将复位并把可程序的部分作为自己的地址。需要注意的是，在通电后设备不要把SDA或SDL线置低，否则将阻碍总线。

——00000100(04H) 由硬件写可程序从地址。所有由硬件定义它们自己可程序部分的设备将接收这2字节时锁住可程序部分。设备将不复位。

——00000000(00H) 不允许这样的码作为第二字节。

程序过程时序发表于有关元件资料上。其余的码还没固定，设备不必响应。当B为1时，这2字节为一般调用。这意味着这2字节由硬件发送，像键盘扫描器一样不能由程序控制发送一个需要的从地址。因为一个硬件主设备预先不知道信息送到哪个设备，它只能产生一般调用并把自己的地址识别告诉系统。

第二字节的前7位是硬件主设备的地址，这个地址由一个联于总线的智能设备如微控制器识别，并指出硬件主设备发出信息的方向。如果这个硬件主设备也作为从设备，则从地址与这个主设备的地址是一样的。

在一些系统中，可能有另外的情况：硬件主设备发送器在系统复位后被设置为从接收器模式。这样，系统配置主设备可知道送地址数据到那个硬件主发送器(现在为从接收器模式)。在这个程序化过程后，这个硬件主设备保持主发送器模式。

2.4.4 START 字节

微控制器可由2种方式联于I2C总线。一个在芯片上有I2C总线硬件接口的微控制器可被程序为只由总线申请中断。当一个设备没有这样的接口时，它只能利用软件经常查讯总线。显然，微控制器监视(或查询)越多，总线为完成预定功能的次数就越少。在快硬件设备与依赖于软查询的相对慢的硬件设备间速度是有差别的。在这种情况下，开始过程先于数据传送，它比在正常情况下占时较长。开始过程包括：

——一个START条件(s)

——一个START字节(00000001)

——确认时钟脉冲(ACK)

——一个重复START条件(Sr)

在一个需要总线访问的主设备发送START条件S后，START字节(00000001)被传送。另一个微控制器以低速率采集SDA线直到START字节的7个0之一被检测到。在SDA线上这个低电平检测到后微控制器变为高速率采集寻找用于同步化的重复START条件Sr。

硬件接收器在接收到重复START条件Sr后复位并不再响应START字节。一个与确认

有关的时钟脉冲在START字节后产生。这代表只与被用在总线上的字节处理格式相一致。任何设备不允许确认START字节。

2.4.5 CBUS 兼容性

CBUS接收器可联于I2C总线，但必须连一DLEN线，省去确认位。一般I2C传送为8位字节的数列。CBUS兼容设备有不同的格式。

在混合总线结构中I2C中的设备不响应CBUS的信息。由于这个原因，非I2C总线兼容设备所响应的一个特殊CBUS地址(0000001x)被保留。在CBUS地址传送后，DLEN线有效，这样即传送CBUS格式。在STOP条件后，所有的设备又可重新接收数据。

在CBUS地址传送后，主传送器可传送CBUS格式。传送由STOP条件终止，并由所有的设备认识。

2.5 I2C 总线的电气特征

I2C元件固定输入电压为1.5V及3V，每个有它自已适当的供给电压。负载电阻必须连于5V±10%供给电压。有相对于Vdd输入电平的I2C总线元件必须有一个公共的电源线，同时也要连一个负载电阻。

当有固定输入电平的元件与输入电平相对于Vdd的元件混合时，后者必须公共连一个5V±10%的电源线，并且必须有负载电阻连于它们的SDA与SCL管脚。

输入电平定义如下：

(1) 在低电平上噪声边界为0.1Vdd

(2) 在高电平上噪声边界为0.2Vdd

(3) 为保护不受由于TV显像管(跳火)在SDA及SCL线上产生的信号尖峰的损害，应串联一个300欧的电阻。

2.5.1 I2C 总线说明的扩展

数据传送率为100Kbit/s和7位寻址已存在有十多年，其说明一直未变。其概念由全世界公认为de facto标准。有上百种不同型号的I2C总线兼容IC从Philips和别的供应者那里合格地生产。现在I2C总线说明的扩展有如下2个特征：

(1) 快模式允许速率增加为4倍。从0到400Kbit/s

(2) 10位的寻址允许使用到1024外加地址

有2个原因使得I2C总线可扩展：

(1) 新的应用要求传送大量串行数据自然要求更高速的位操作。改进的IC生产技术使得在不增加接口电路生产成本的前提下增加4倍的速度。

(2) 多数由7位寻址的112地址方案多次发布。为防止由于增加新元件而分配从地址带来的 问题，最理想的办法是采用多地址结合。由于新的10位选址大约增加10倍的有效地址。

I2C总线的所有新元件都提供快模式。它们能够接收与传送速率达400Kbit/s。最低要求是它们能与400Kbit/s同步，延长SCL的低(LOW)周期以减慢传送。快模式元件必须能向下兼容，这意味着它们必须仍然能与速度从0到100Kbit/s的元件在0以100Kbit/sI2C总线系

统中通讯。

显然，0—100Kbit/s 的 I2C 元件不与用于快模式 I2C 总线系统。因为它们不能适应更高的传送速率。否则不可预见的状态在这种元件中会发生。

具有快模式 I2C 接口的从设备有 7 位或 10 位地址，使用 7 位地址最节省硬件和具有最短的信息长度。7 位与 10 位地址元件可混全一起，无论是 0 到 100Kbit/s 的标准模式还是 0 到 400Kbit/s 的快模式。目前的或将来的主设备可产生 7 位或 10 位地址。

2.5.2 快模式

在 I2C 总线的快模式中以前 I2C 总线说明中的协议、格式、逻辑电平及 SDA，SCL 的最大负载能力都保持不变。对以前 I2C 总线说明中的改变部分如下：

(1) 最大位速率增加到 400Kbit/s。

(2) 串行数据及串行时钟信号已改变。无须与别的总线系统像 CBUS 兼容，因为它们不能在增加的位速率下运行。

(3) 快模式的输入元件必须有信号尖峰抑制和在 SDA 及 SCL 输入时有一施密特触发器。

(4) 快模式元件的输出必须有 SDA 及 SCL 信号下降沿的顷斜控制。

(5) 如果电源供应关闭，SDA,SCL I/O 管脚必须悬浮以不阻碍总线。

(6) 联于总线的外部负载元件必须修改以适应较短的快模式 I2C 总线的最大允许回升时间若总线装有 200pf，则负载元件对每个总线连线而言为一个电阻，若总线装 200pf 和 400pf，则负载元件为一个电流源(3mA max)或一个开关电阻电路。

2.6 10 位寻址

在 I2C 总线说明中 10 位寻址不改变格式。使用 10 位寻址要启用 8.1 节中所述的在 START(S)或重复 START(Sr)条件后的第一个字节中的保留位 1111XXX。10 位寻址不影响现存的 7 位寻址。7 位和 10 位地址元件可联于同一 I2C 总线，寻址既可用于标准模式系统(0 到 100Kbit/s)，又可用于快模式系统(0 到 400Kbit/s)。

虽然保留地址的位 1111XXX 有 8 种可能的组合，只有 4 种组合用于 10 位寻址。其它的组合 11111XX 留作将来 I2C 总线的增强。

2.6.1 在前 2 个字节中位的定义

在 START 条件(S)或重复 START 条件(Sr)后 10 位从地址由前 2 个字节形成。

第一个字节的前 7 位与 11110XX 组合，XX 表示为 10 位地址的高 2 位字节(MSBs)，第一个字节的第 8 位为 R/W 它决定信息的传送方向。第一个子字节中的第 8 位为 0，表示主设备写信息到选中的从设备，1 表示将从从设备读信息。

如果 R/W 为 0，则第二个字节含有 10 位地址的其余 8 位，如果 R/W 为 1，则下一个字节表示从从设备到主设备传送的数据。

2.6.2 10位地址格式

在传送中读/写格式有多种组合，其中包括10位寻址。可能的格式如下：

(1) 主发送器发送到有10位从地址的从接收器，传送方向不变。在START条件后发一个10位地址，每个从设备与自已的地址比较第一个字节的前7位从地址(11110XX)并检测第8位(R/W方向位)。可能有不止一个的设备再把二个字节的8位从地址(XXXXXXXX)与已的地址比较，但只有一个相配并产生确认(A2)。相配的从设备将保持这个由主设备选中的地址，直到它收到一个STOP条件(P)或在一个不同的从地址后一个重复START条件(Sr)。

(2)主接收器从一个有10位从地址的从发送器读。在第二个R/W位后传送方向改变。其过程同(1)中所述的主发射器寻址一个从接收器从开始直到并包括确认位A2的过程一样。在重复START条件(Sr)后一个相配的从设备因被寻址过，这个设备就检查Sr后的第一个字节的前7位是否与START条件(S)后的一样，并检验第8位(R/W)是否为1。如果这些都 相配，这个从设备认为自已被寻址为发送器并产生一个确认A3。从发关器保持这个地址直到它接收到一个STOP条件(P)或在一个不同的从地址后的重复START条件(Sr)。在Sr后所有别的从主设备总是占有总线。

(3)组合格式。10位与7位在串行传送中的组合。

在每一个START条件(S)或每一个重复START条件(Sr)后，10位及7位从地址被发送。

2.6.3 一般调用地址及START字节

在I2C总线的10位寻址过程中，START条件(S)后的2个字节通常决定哪一个从设备被选中。一个例外是一般调用地址为00000000(H′00′)。从设备的10位寻址对一般调用的反应同7位寻址一样(参看8.1.1)。

在一般调用后，硬件主设备可发送它们的10位地址。在这种情况下，一般调用地址字节后跟2个含有主发送器10位地址的连续字节。

同7位寻址方式一样，START字节00000001(H′01′)先于10位寻址(参看2.4.4)。

第三章　有关集成电路

GRAND VIDEO 采用了 Chip & Technologies 公司的 82C9001 和 Philips 公司的视频芯片集。按其复杂程度和是否可编程，我们分别作全面或简单介绍。

3.1　82C9001 PC 视频窗控制器说明

3.1.1　一般介绍

活动视频图像在计算机图形显示器上显示的扫描速率变换和窗口控制。

窗口位置可由独立的 X，Y 座标和彩色键控制。

图像可以独立地在 X 和 Y 方向压缩(标定)一直到原图的 1/64。

停帧俘获及真彩色显示。

输入分辨率在全广播质量视频带宽情况下可达 1024(水平) * 512(垂直)像素。

显示分辨率可到 800 * 600。

支持来自工业标准数字化芯片的 NTSC，PAL，SECAM，S－VHS 和 RGB 诸输入格式。

支持标准的 4:1:1，4:2:2YUV 和 16 位 RGB 格式。

支持内存高效的 2:1:1 YUV 格式。

隔行扫描或逐行扫描视频输入。

支持隔行扫描或逐行扫描输出。

输出放大因子为 2，4 或 8。

对于带 82C457 的面板显示支持完全的运动彩色视频。

3.1.2　视频格式

计算机图形的彩色像由 R，G，B 控制。视频与 TV 的图像由一个亮度(Y)和两个色度(U，V)组成的信号决定。亮度带宽是每个色度带宽的两倍。由于容易把 Y，U，V 复合成视频信号或把它们分离出来，数字化视频系统往往采用亮度/色度方式。采用这种方式表示像比用 R，G，B 方式更能有效地利用内存。一个 640×480 的像由 16 位 R，G，B 方式产生 65536 个彩色(点)需要 1 兆 VRAM，而同样的像由 Y，U，V 坐标存储只占用 768K VRAM 并可产生 2 百万个彩色(点)。

PC 视频的可变性允许它既支持 R，G，B 方式又支持 Y，U，V 方式。它所支持的主要 4 种格式如下(这里 X:Y:Z 指三个视频分量)：

(1) 4:1:1

亮度(Y)每个像素采一次，色度分量(U，V 或 Yr，Yb)由 4 个像素平均。因此，每输入 4 个像素就要有 4 个亮度，一个 U 和 V，共 6 个值。

(2) 4:2:2

在格式 4:1:1 基础上加大了彩色宽度。色度分量是 2 个像素的平均。每 4 个像素要有 4

个亮度，两个色度(U 和 V 各两个)，共 8 个值。这是国际广播标准。

(3) 2:1:1

这是最省内存的格式。每两个像素一个亮度，每 4 个像素一个色度(U,V 各一个)。因此，4 个像素共需 4 值。

(4) 4:4:4

这种格式一般用于 R,G,B 方式，但也用于 Y,U,V 方式。Y,U,V 每个像素各采一次，因此 4 个像素需 12 个值。每个分量上的带宽是一样的。

3.1.3 信号流

在 PC 视频子系统中，图像采集和显示是两个独立的可程序过程。并且使用不同的时钟。

3.1.4 采集

PC 视频提供控制信号到一个标准的数字视频芯片，采集窗口坐标是可程序的，使得能够对输入图像修剪或平动。数字视频通过 PC 视频内部的 FIFO 存储在图形缓冲器中。采集过程与输入视频 VSYNC 及 HBLANK 信号同步，并以视频采集速率执行。图像采集数据送到双口视频 RAM 的随机入口，作为采集过程的一部分，对输入视频数据完成标定。标定的图像存在图像缓冲器中。

当采集 PAL 制视频时，当前 576 行的最大 512 行由输入修剪窗口或输入标定采集和选取。通过输入标定按一定比例保存的整个 PAL 画面可在 VGA 上显示出来。输入窗口既支持隔行扫描视频(缺省)又支持逐行扫描视频。以下 4 种俘获模式都有效：

(1) 隔行扫描　　(2) 偶场　　(3) 奇场　　(4) 逐行扫描

在非标定隔行扫描模式下，光栅地址每次增加 2。偶场写偶行的数，奇场写奇行的数。偶场从 0 开始每次加 2：0，2，4，……。奇场从 1 开始，每次加 2：1，3，5，……，直到最大行的数目。

地址多路为采集窗口，视频显示窗口或主处理器选择地址。不论活动视频采集还是 CPU 读写访问都可进行，但二者不同时发生。

3.1.5 显示

显示过程通过特征连接器与 VGA，HSYNC，VSYNC，BLANK 及 CLOCK 信号同步。数据从视频 RAM 的串行口输出。视频图像可在 VGA 显示屏上的矩形窗口内或外显示。

3.1.6 系统时钟

视频 RAM 控制及时钟逻辑使用双比率输入时钟。双时钟比率 27MHz 为行锁，在由视频磁带，静态视频摄像机和以非时间为基础的稳定源输入视频同步时跟踪磁差。最大输入时钟比率是 30MHz。

3.1.7 CPU 接口

PC 视频有一个非多路的 24 位地址总线和一个 16 位的数据总线。使用 AT 总线信号 ALE PC 视频锁栓 AT 总线地址 LA[20:23]。只有当 PC 视频 I/O 寄存器使用 8 位周期访问

时，PC 视频存储器才可使用 8 位和 16 位总线周期访问。PC 视频控制内存访问的所有字节交换。在 PC 地址空间中当内存映射 1M 以上字节时，PC 视频支持 16 位内存周期并产生 MEM-CS16/。以 PC 视频为基础的板可插到 16 位总线槽上。

PC 视频可被程序既响应固定的 I/O 地址又响应软件编程的 I/O 地址。当总线 RESET 信号从高到低时，CS/管脚的电平由 PC 视频读取。如果 CS/是有效状态即低电平，PC 视频响应 I/O 地址 0AD6－7H，当 CS/为无效即高电平时，它使用程序给的 I/O 地址寄存器位 7－0 检查有效 I/O 地址空间。当选用这种模式时，由程序给定的 I/O 地址寄存器的各位由第一 CPU I/O 写周器和 CS/有效写入。因为 PC 视频总是占用两个连续 I/O 地址，这个寄存器的 0 位是不用的。

3.1.8 内存接口

内存由 2 个存区组成宽 1024，高 512(1024×512)。内存深度或是 12 位 4:1:1 多路数据格式，或是 16 位 4:2:2 多路数据格式和 4:4:4 非多路数据格式。

PC 视频使用 256K×4，100ns VRAM 支持 3 种 VRAM 配置：(1) 2 个分区的 VRAM，3 个 VRAM 在分区 0，一个用于色度使用的 VRAM 在分区 1。(2) 2 个分区的 VRAM，每个分区 3 个 VRAM。(3) 2 个分区的 VRAM，每个分区 4 个 VRAM。PC 视频也支持 VRAM 写位的屏蔽功能。帧缓冲存储器作为线性地址空间是可访问的，它大于 1024K(100000H)。视频帧缓冲存储器可由字节或字周器访问。PC 视频产生一个等待状态供正常内存访问使用，当内存访问与 VRAM 刷新周期或数据传送相冲突时，则产生更多的等待时间。当 PC 视频内存被 CPU 访问时，视频俘获必须由软件先于访问暂停。PC 视频内存在 RESET 时不可访问，当访问时，必须明显使之使能。

3.1.9 中断支持

PC 视频支持 CPU 响应视频同步 Vsync。中断源被保留，且可由软件清除。

3.1.10 视频 RAM 时序发生器

同步时序发生器管理来自双比率视频数据时钟的视频 RAM 控制信号。视频 RAM 有 2 个工作区，每个区都是"快页"模式。每区接收一行像素，两区交替接收，使用公共总线，驱动像素比率为 13.5MHz。每区的列地址信号(CAS)或使能选通包含在像素数据中。

内部刷新在输入水平视频的间隙进行。处理器访问被限制在当采集窗口冻结和页模式写周期无效时进行。CPU 访问使用标准的读/写周期，采用并行数据路径，旁路输入视频 FIFO。

VGA 显示窗口使用 1M(256×4) VRAM 串行移位寄存器扫描显示数据。VRAM 串行输出只能用作读模式，VRAM 的两个区分别由两个 512 像素 VRAM 移位寄存器以乒乓格式数据结构提供每行 1024 个像素的数据。

支持下列 VRAM 内存周期：

周期类型	功能
(1) 随机读/写周期	处理器访问
(2) RAS 刷新前 CAS	仅刷新
(3) 移位寄存器装载	为输出显示装载移位寄存器

(4) 快页模式写　　　　视频采集
(5) 写位屏蔽　　　　　在视频采集或 CPU 写期间数据位屏蔽
(6) 移位寄存器输出　　时钟为 1.5 输出显示比率

3.1.11 标定

支持两个独立输入标定比率：一个是用于水平，另一个是用于垂直，垂直比率用于隔行扫描。标定由水平地减少像素及垂直方向减少行来实现。独立的标定因子范围从 1/64 到 63/64，水平与垂直方向都如此。把标定关掉则标定因子为 64/64。

3.1.12 显示窗口顶层

模拟 VGA 与模拟视频数据的混合由外部模拟多路器完成。多路转接控制信号可由三种不同的方法产生：(1) 在 PC 视频 X—Y 窗口控制寄存器中定义一个输出窗口；(2) 键控 VGA 颜色(颜色比较)；(3) 在显示内存的 LSB 中写入一个模式。在(1)和(2)的情形下多路转接器的功能由 PC 视频产生，此时使用 VGA 时钟(PCLK)。在(3)的情形下显示内存的 LSB 由外部门控，以控制模拟多路转接器。

3.1.13 X—Y 窗口控制寄存器

X—Y 窗口区域由 4 个寄存器定义。4 个寄存器分别用 X 初始点，X 结束点，Y 初始点，Y 结束点来表示一个矩形区。X—Y 窗口位置取决于 VGA 同步信号，其大小由 VGA 时钟及行决定。

3.1.14 彩色控制键

控制信号由 8 位 VGA TTL 视频数据(P7—0)与彩色键中的 8 位值比较产生。要同时支持一种以上的颜色，就要有 8 位 VGA 数据屏蔽寄存器。这个屏蔽寄存器删除那些把 1 写入 VGA 数据寄存器中相应位位置的数据位。VGA 像素数据由 PCLK 采集。

3.1.15 显示数据的 LSB

内存显示的 LSB 可被用来控制模拟多路。PC 视频支持 VRAM 每位写屏蔽写功能。软件可在显示内存的 LSB 写入一个模示并用屏蔽保护键控模示。内存显示的 LSB 用来控制在像素基线上对一个像素的模拟多路。因为内存显示 LSB 写保护，视频采集不修改这一行。每位写屏蔽可以用于 16 位宽数据线路的任何位。由于亮度数据对 4:1:1 输入只有 7 位宽，一般使用亮度数据的 LSB。

3.1.16 复盖功能

显示窗口和彩色键区域有 4 种结合的可能，它们是：非窗口和非键区(F0)，有窗口区无键(F1)，有键区无窗口(F2)，和既有窗口区又有键区(F3)。对像素基线上的一个像素这 4 个功能选择自己的功能位。如果一个功能位所对应这个区域是 0，则该区域显示 VGA 图形，若为 1，则视频输出数据复盖于 VGA 图形的上面。如果 X—Y 窗口无效，区 F1 及 F3 将不存在。如果键控无效，则区 F2 及 F3 将不存在。

3.1.17 显示区域平移

显示区域中的视频数据可水平扫描1024个像素，其步长是：在4:1:1输入Y/C格式中为4个像素，在4:2:2输入Y/C格式中为2个像素。视频数据可被垂直扫描511行，每次扫描1行。

3.1.18 PC视频的整体使能与禁止

PC视频I/O寄存器索引FFH为PC视频使能与禁止而保留。这是一个只写寄存器，其0位用来控制寄存器访问，只有RESET才能清除这一位，当PC视频的I/O地址确定后，这位应为1以使能访问别的PC视频I/O寄存器。当这个寄存器的1位设置为1时，PC视频帧缓冲器是可访问的。这个寄存器的高4位是可读的，它们反映PC视频的版本/部分数。

管脚分布和功能

共有4*40个管脚，名称及分布参看图3.1，其管脚说明如下：

管脚6—28(A0—A19)：	输入，高有效；20位系统地址总线。
2—5(LA20—23)：	输入，高有效；系统地址总线的高4位。它们在ALE的下降沿被锁闩。
52—59(D0—7)和 43—50(D8—15)：	输入/输出，高有效；系统数据总线。
65(RESET)：	输入，高有效；复位输入。
34(RFSH/)：	输入，低有效；系统刷新。当这个输入为低时，当前的存储周期不被理会。
61(ENLO/)和51(ENHI/)：	输出，低有效；使能对于数据总线的外部数据收发器的控制。
62(DIR)：	外部数据总线收发器的方向控制：0，从PC Video中读；1，写入PC Video。
29(BHE/)：	输入，低有效；使能16位接口。低标志着当前字地址的高阶正在被计取。配合A0使用，标明哪些字节被传送到总线(所有字节导向在内部完成)。

BHE/	A0	作用
0	0	两字节D15:0
0	1	高字节D15:8
1	0	低字节D7:0

32(AEN)：	输入，高有效；定义有效I/O地址：0，有效I/O地址，1，无效I/O地址(DMA周期)。如果使用单周期DMA，内存地址将在IORD/或IOWR有效的同时到总线。 在AEN=1时，PC Video将不响应IORD/或IOWR/。
31(ALE)：	输入，高有效；地址锁闩使能。此信号下将沿锁闩系统地址。

图 3.1　82C9001 芯片管脚图

33 (RDY)：　输出，高有效；准备好。低表明当前周期与等状态一起被延长，高表明已准备好，然后处于三态。此信号正常情况下为三态，只有当 PC Video 不能立刻响应一个存储请求时才为低，并且有等状态。

71 (MWMCS16/)：　输出，低有效；标志 16 位内存周期。由 PC Video 断言以表明芯片有能力对请求的地址把 16 位传送到总线。

37 (MEMR/)：　输入，低有效；表明存储读周期从 AT 总线。

38 (MEMW/)：输入，低有效；表明存储写周期从 AT 总线。

35 (IORD/)：输入，低有效；表明 I/O 读周期。

36 (IOWR/)：输入，低有效；表明 I/O 写周期。

63 (IRQ/)：输出，高有效；视频 VSYNC 中断。当中断已被使能但没有中断悬起时此管脚为低，当中断已被使能有一个中断悬起时此管脚为高。

131—138 (YD7—0)：输入/输出，高有效；亮度数据总线。它们联到 VRAM 数据管脚。

123—130 (CD7—0)：输入/输出，高有效；色度数据总线。它们联到 VRAM 数据管脚。

150—158 (MA8—0)：输出，高有效；存储器的区和面 VRAM 地址总线。

141 (RASY/)：输出，低有效；为亮度平面 VRAM 行地址选通。

139 (RASC/)：输出，低有效；为色度平面 VRAM 行地址选通。

142 (CAS1/)和 143 (CAS0)：输出，低有效；VRAM 列地址选通。存储器的每个区都有一个选通脉冲。

146 (WE1/)和 147 (WE0)：输出，低有效；VRAM 写使能。存储器的每个区都有一个使能。

145(DTOY/)和 Pin144(DTOC)：输出，低有效；数据传送/输出使能。存储器的每个区都有一个控制。

148(SC)：输出，高有效；所有 VRAM 的移位时钟。这个信号也用来多路传送来自 VRAM 区的输出数据。

86—93 (P0—7)：输入，高有效；来自 VGA 特征联结器的像素数据。这个数据用于彩色匹配以控制 VGA 图形和视频输出的覆盖。

85 (PCLK)：输入，高有效；VGA 像素时钟。该时钟用于闩 VGA 像素数据和同步视频复盖窗口。

84 (HSYNC)和 82 (VSYNC)：输入，高有效；83 (BLANK/)：输入，低有效；此三个管脚分别为来自 VGA 特征联结器的水平同步，垂直同步和空白信号。这些信号用来同步视频和 VGA 图形的复盖。

72 (VFLD)：输入，高有效；外部场位。当被选取后，这个信号识别隔行扫描输出视频数据流的奇/偶场顺序。

107—115 (BY0—7)：输入，高有效；来自数字视频源的亮度数据。这个数据被缓冲并储存于用于视频输出的 VRAM。

97 (BC0)，98 (BC1)，94 (BC2)，102—106 (BC3—BC7)：输入，高有效；来自数字视频源的色度数据。这个数据被缓冲并储存于用于视频输出的 VRAM。

114 (XCLK)：输入，高有效；视频数据时钟。这个时钟用来闩来自数字视频源的视频数据。

118 (PCX2)：	输入，高有效；两倍视频数据时钟。用于产生 VRAM 时序。
119 (XVSY)：	输入，高有效；来自数字视频源的垂直同步信号，它用于视频数据的存储。
116 (XHBL)：	输入，高有效；水平空白。它用于识别数字视频输出数据流中的有效区域。
117 (XFLD)：	输入，高有效；外部场位。PC 视频使用内部产生的场信号，作为缺省。如果需要，可由设置寄存器 21 的位 6 为 1(高)的办法选取这个管脚的场信号。
30 (CS/)：	输入，低有效；片选(I/O)。该管脚由复位的下降沿采取。如果为低，PC 视频将在单元 0AD6 和 0AD7 响应 I/O 访问。如果为高，则被编程的 I/O 寄存器和 CS/输入用来检测有效 I/O 地址。
96 (I2CK)和 95(I2CO)：	输出，高有效；74(I2CI)：输入；这些开集 I/O 管脚被设计用来支持 I2C 总线。
78 (DICL)：	输入，高有效；显示时钟输入。这个输入提供用于输出视频数据的时钟。
76 (DACL)：	输出，高有效；视频 DAC 时钟。该信号由 DICL 导出，同步输出视频数据到视频 DAC。
75 (NDBL/)：	输出，低有效；视频空白输出到 DAC。这个信号与 DICL 同步并使来自 VRAM 存储器的有效视频数据合格。
77 (KEYO)：	输出，低有效；视频/VGA 多路。当高时，输出数据多路选择视频数据，当低时，输出为 VGA 图形。
66－69 (GP3－0)：	输出，高有效；一般目的 I/O 和选通脉冲。这些管脚可以作为被定义为外部计存器的附加用户的 I/O 选通。把计存器 18 的位 5 设置为 0(低)则可定义 GP01 为锁相回路。
1,20,40,81,100,120 (VCC)：	＋5V 电源
21,41,42,60,79,80,101,121,122,140,159,160 (GND)：	地
70,64,149,73(TST0－3)：	输出，高有效；检查管脚。这些管脚应悬而不接。
19,39,99 (Reserved)：	输出，高有效；保留。这些管脚应悬而不接。

以下就其功能分类介绍：

3.2 寄存器

3.2.1 寄存器寻址

PC 视频控制寄存器地址定位由 CS/在 RESET 时的输入状态决定。如果在 RESET 下降

沿时 CS/输入是低，则 PC 视频响应固定的 I/O 地址:0AD6－7H(对一个有效的 I/O 译码，CS/必须为低)。如果 CS/为高，则使用可编程地址。当前数据总线上的值在紧接着 RESET 周期后的第一个有效 I/O 写(CS/＝0，IOWR/＝0)时被装入程序 I/O 地址寄存器的 7－0 位，这个值被用作 PC 视频寄存器的 I/O 地址定位。因为 PC 视频总是占用 2 个连续的 I/O，这个寄存器的 0 位起作用。I/O 地址的低 8 位由程序寄存器中的值决定。其余地址位取决于产生 CS/选通脉冲的外部译码逻辑。

用于 PC 视频芯片的设置与控制寄存器占用 2 个连续地址单元。第一个或偶单元包含一个索引寄存器。索引寄存器决定或指出所有别的寄存器。第二个或奇单元为选择寄存器的数据口。访问 PC 视频寄存器时，把要访的索引值写入索引寄存器。这个寄存器的数据被读或写在数据寄存器单元。

在 PC 视频芯片中有 4 组控制寄存器。它们是：CPU 接口，通用 I/O 控制，视频采集和显示窗口控制。一旦 PC 视频由整体使能寄存器的 0 位使能，所有这些寄存器为可读/写。

3.2.2 CPU 接口寄存器

这些寄存器分别用来使能/禁止 PC 视频，存放 I/O 地址定位，指明内存地址和配置，设置缓冲内存写屏蔽和中断服务。在别的寄存器被读或写之前，整体使能寄存器(索引:FFH)的 0 位必须置 1，此位在设置之前该寄存器和索引寄存器为只读状态。

3.2.3 通用 I/O 寄存器

这些寄存器控制通用 I/O 管脚和 I2C 总线接口。

3.2.4 视频采集寄存器

这些寄存器用来控制采集活动视频数据流。它们提供当前采集状态数据，控制输入采集窗口，标定和存放内存定位。

3.2.5 显示窗口寄存器

这些寄存器用来控制输出视频数据流和颜色键。它们提供窗口定位和大小的控制，数据起始定位(平动)，移位时钟，模拟多路偏斜，VGA 彩色键和屏蔽。

索引寄存器(RX)：读/写

位 0－7：索引值用来访问控制寄存器

I/O 地址寄存器(R00)：读/写，索引:00H

位 0：保留

1－7：与地址输入的 A7－A1 比较以检出有效 I/O 地址。在 RESET 时，如果 CS/为低，这个寄存器 被初始化到 D6H，如果 CS/为高，则第一个 I/O 写时(IOWR/＝0,CS/＝0)把数据输入(D7－D1)的当前值装入这个寄存器。

内存访问寄存器(R01)：读/写，索引:01H

位 0－3：保留(0)

4：使能 VRAM 写屏蔽：0 禁止，1 使能。

5—7：保留(0)

线性内存基本地址寄存器(R06)：读/写，索引：06H

位 0—3：定义为线性内存空间的起始地址。地址分辨力为 1 兆。

4—7：保留(位 4 应设为 1)

数据屏蔽寄存器，亮度数据(R07)：读/写，索引：07H

位 0—7：控制 VRAM 对亮度数据的每位写特征。若某位为 0，则该位数据在视频数据采集时被保护不被修改。这个寄存器由 R01 的位 4 使能。

数据屏蔽寄存器，色度数据(R08)：读/写，索引：08H

位 0—7：控制 VRAM 对色度数据的每位写特征。若某位为 0，则该位数据在视频数据采集时被保护不被修改。这个寄存器由 R01 的位 4 使能。

中断屏蔽寄存器(R09)：读/写，索引：09H

位 0：视频偶垂直同步中断使能。0 为禁止，1 为使能。

1：视频奇垂直同步中断使能。0 为禁止，1 为使能。

2—5：被轮询以监视信号状态，位 2：查询视频垂直同步，位 3：查询视频场(0：偶，1：奇)，位 4 查询 VGA 垂直同步，位 5：查询 VGA 水平同步。

6—7：保留(0)

通用 I/O 寄存器 0(R10)：I/O，索引：10H

通用 I/O 寄存器 1(R11)：I/O，索引：11H

通用 I/O 寄存器 2(R12)：I/O，索引：12H

通用 I/O 寄存器 3(R13)：I/O，索引：13H

这 4 个寄存器通过外部锁存和/或缓冲执行。如果一个通用 I/O 寄存器使能，则当这个寄存器读或写时其选通脉冲由适当的 GP I/O 管脚产生。作为需要，外部逻辑对于锁存系统总线数据或把数据驱动到系统总线是必要的。数据收发控制信号对这些 I/O 周期是有效的。

通用 I/O 控制寄存器(R18)：I/O，索引：18H

位 0：I2C 总线时钟

1：I2C 总线数据

位 0 和位 1 设计成用于控制 I2C 总线，分别联于它们相应的 I2C 输出管脚，输出为开路集电集信号。

2：I2C 总线读回输入管脚 I2CI，这个管脚应联于 I2CO。该位反映 I2CK 管脚从 0 升到

1 时的 I2CI 状态。

3：保留(0)

4：通用 I/O 0。0 在 GPIO0 上输出 R10 的译码，1 保留。

5：通用 I/O 1。0 在 GPIO1 上输出"PLLHREF"，1 在 GPIO1 上输出 R11 的译码。

6：通用 I/O 2。0 在 GPIO2 上输出 R12 的译码，1 保留

7：通用 I/O 3。0 在 GPIO3 上输出 R13 的译码，1 保留

注:为正确操作,位 4,6,7 应由程序设为 0

视频采集模式寄存器(R20)：读/写，索引：20H

位 0：开始视频采集。0 停止视频采集并容许 CPU 访问帧缓冲器，1 开始视频采集。采集类型如连续/信号帧,隔行扫描等由这个寄存器的 1—3 位决定。该位由硬件在视频垂直同步周器的最后更新。软件必须在停止视频采集后把该位读回[READ BACK]为 0 以保证访问帧缓冲器。

1：信号/连续。0 连续视频采集，1 需要一个信号场或帧(由位 2,3 决定)。这个寄存器的 0 位在场或帧的最后被清除。

2：场/帧。0 需要输入视频帧，1 需要输入视频场(只对隔行扫描模式)。

3：奇/偶。0 需要一个偶(第一)场，1 需要一个奇(第二)场。

4：水平同步极性。0 视频水平输入为低有效，1 视频水平输入为高有效。

6：保留(0)

7：隔行扫描。0 输入视频为隔行扫描，1 输入视频为非隔行扫描。

采集窗口控制寄存器(R21)：读/写，索引：21H

位 0：视频输入修剪。0 禁止，1 使能。

1：视频俘获。0 在修剪窗口内俘获视频，1 在修剪窗口外俘获视频。

2：视频输入标定，水平方向。0 禁止，1 使能。

3：视频输入标定，垂直方向。0 禁止，1 使能。

4：视频输入数据多路。这个位决定视频输入数据为多路的还是非多路的，即 YUV 或 RGB。

0 多路(YUV)，1 非多路(RGB)。

5：多路比率。这个位决定亮度亮度及色度输入比率,只有当位 4 为 0 时才有效。0：4:1:1,1：4:2:2。

6：选择外部场。0 内部场，这个场在 XVSYNC 输入管脚的 XCLK 的下降沿被检测到。1 场位通过 XFLD 管脚输入，并在 XFLD 使用前由 XCLK 重定时钟。在 XVSYNC 的下降沿 XFLD 应变换。XFLD 的 0 表示"偶场"，1 表示"奇场"。

7：反转场。内部或外部产生场位的极性反转。0 场位不被修改，1 场极性反转。采集窗口，X 开始低位寄存器(R22)：读/写，索引：22H。

0—7：为一个 10 位寄存器的低 8 位。其值定义水平视频采集窗口的开始。这个值由输入像素时钟检测并由视频水平的下降沿选取。

采集窗口，X 开始高位寄存器(R23)：读/写，索引：23H

位 0—1：为一个 10 位寄存器的高 2 位。其值定义水平视频采集窗口的开始。这个值由输入像素时钟检测并由视频水平的下降沿选取。

2—7：保留(0)

采集窗口，Y 开始低位寄存器(R24)：读/写，索引：24H

位 0—7：为一个 10 位寄存器的低 8 位。其值定义垂直视频采集窗口的开始。这个值由输入行检测并由垂直同步及垂直开始调整视频的下降沿选取。

采集窗口，Y 开始高位寄存器(R25)：读/写，索引：25H

位 0—1：为一个 10 位寄存器的高 2 位。其值定义垂直视频采集窗口的开始。这个值由输入行检测并由垂直同步及垂直开始调整视频的下降沿选取。

2—7：保留(0)

采集窗口，X 结束低位寄存器(R26)：读/写，索引：26H

位 0—7：为一个 10 位寄存器的低 8 位。其值定义水平视频采集窗口的结束。这个值由输入像素时钟检测并由视频水平的下降沿选取。

采集窗口，X 结束高位寄存器(R27)：读/写，索引：27H

位 0—1：为一个 10 位寄存器的高 2 位。其值定义水平视频采集窗口的结束。这个值由输入像素时钟检测并由视频水平的下降沿选取。

2—7：保留(0)

采集窗口，Y 结束低位寄存器(R28)：读/写，索引：28H

位 0—7：为一个 10 位寄存器的低 8 位。其值定义垂直视频采集窗口的结束。这个值由输入行检测并由垂直同步及垂直开始调整视频的下降沿选取。

采集窗口，Y 结束高位寄存器(R29)：读/写，索引：29H

位 0—1：为一个 10 位寄存器的高 2 位。其值定义垂直视频采集窗口的结束。这个值由输入行检测并由垂直同步及垂直开始调整视频的下降沿选取。

2—7：保留(0)

采集地址低寄存器(R2A)：读/写，索引：2AH

位 0—7：为一个 20 位指针的低 8 位。其值指向帧内存定位，视频采集从这里开始。这是一个线性地址，在一个视频行的结尾，地址重新设置为这个行的开始并加上 1024 字节的偏移构成下一行的起始地址。

采集地址中寄存器(R2B)：读/写，索引：2BH

位 0—7：为一个 20 位指针的低 8 位。其值指向帧内存定位，视频采集从这里开始。这是

一个线性地址，在一个视频行的结尾，地址重新设置为这个行的开始并加上1024字节的偏移构成下一行的起始地址。

采集地址高寄存器(R2C)：读/写，索引：2CH

位0—7：为一个20位指针的高4位。其值指向帧内存定位，视频采集从这里开始。这是一个线性地址，在一个视频行的结尾，地址重新设置为这个行的开始并加上1024字节的偏移构成下一行的起始地址。

采集水平标定寄存器(R2D)：读/写，索引：2DH

位0—5：这些位定义每输入64个像素所写的像素数。有效值为1—63。水平标定通过R21的位2禁止。

6—7：保留(0)

采集垂直标定寄存器(R2E)：读/写，索引：2EH

位0—6：这些位定义每输入64个行写的行数。有效值为1—63。垂直标定通过R21的位3禁止。

7：保留(0)

标定场调整寄存器(R2F)：读/写，索引：2FH

位0—6：在采集时为奇场修改标定值。这是一个诊断寄存器，其值应设置与标定寄存器R2E在正常操作时的值一样。

7：保留

输入视频起始调整(R30)：读/写，索引：30H

位0—5：说明从视频垂直同步到现行视频帧开始的扫描数。该寄存器总是由程序设定为非0值。

6—7：保留

标定控制寄存器(R38)：读/写，索引：38H

位0—1：色度多路调整位。这2位提供保持亮度/色度校准的调整。

2：Y重写模式。此位与小于1/2的垂直标定一起使用以减少在场际运动中由移动图像引起的运动的人为因素。当标定小于1/2时，场俘获与该位应置为1，这导致只从一个视频场中写一个有比例的图像。0正常标定，1修改的标定。

3：X最大使能。此位防止内存中X地址环绕[WRAP AROUND OF ...]，0禁止，1使能.

4：Y最大使能。此位防止内存中Y地址环绕[WRAP AROUND OF ...]，0禁止，1使能对PAL视频数据此位应使能。

5：保留。对正常操作此位应置为0

6：保留。对正常操作此位应置为0

7：快写使能。当此位置 1 时，要求 CPURDY 早一个时钟以改善 CPU 内存写周期的速度。在 RESET 时此位缺省置为 0。

显示区域控制寄存器(R40)：读/写，索引：40H

位 0：使用 X—Y 窗口，窗口复盖。0 禁止，1 使能。

1：使用彩色键，窗口复盖。0 禁止，1 使能。

2：非彩色键或 X—Y 窗口区域(功能 0)。0 显示 VGA，1 显示帧缓冲器数据。

3：X—Y 窗口区域(功能 1)。如果这个寄存器的 0 位为 0 则这个区域不存在。0 显示 VGA，1 显示帧缓冲器数据。

4：彩色键区域(功能 2)。如果这个寄存器的 1 位为 0 则这个区域不存在。0 显示 VGA，1 显示帧缓冲器数据。

5：既 X—Y 窗口区域又彩色键区域(功能 3)。如果这个寄存器的 0 位与 1 位都为 0 则这个区域不存在。0 显示 VGA，1 显示帧缓冲器数据。

6—7：这 2 位定义 VGA 数据输入与在 VGA 时钟多路控制输出间的偏斜[SKEW]。

00 延时 2 VGA 时钟

01 延时 3 VGA 时钟

10 延时 4 VGA 时钟

11 延时 5 VGA 时钟

显示窗口，X 开始低字节寄存器(R41)：读/写，索引：41H

位 0—7：为一个 11 位寄存器的低 8 位。其值定义窗口的水平开始。这个值由 VGA 像素时钟检测并由 VGA 水平同步的下降沿选取。

显示窗口，X 开始高字节寄存器(R42)：读/写，索引：42H

位 0—2：为一个 11 位寄存器的高 2 位。其值定义窗口的水平开始。这个值由 VGA 像素时钟检测并由 VGA 水平同步的下降沿选取。

3—7：保留(0)

显示窗口，Y 开始低字节寄存器(R43)：读/写，索引：43H

位 0—7：为一个 10 位寄存器的低 8 位。其值定义窗口的垂直开始。这个值由 VGA 行检测并由 VGA 垂直同步的下降沿选取。

显示窗口，Y 开始高字节寄存器(R44)：读/写，索引：44H

位 0—1：为一个 10 位寄存器的高 2 位。其值定义窗口的垂直开始。这个值由 VGA 行检测并由 VGA 垂直同步的下降沿选取。

2—7：保留(0)

显示窗口，X 结束低字节寄存器(R45)：读/写，索引：45H

位 0—7：为一个 11 位寄存器的低 8 位。其值定义窗口的水平结束。这个值由 VGA 像素

时钟检测并由 VGA 水平同步的下降沿选取。

显示窗口，X 结束高字节寄存器(R46)：读/写，索引：46H

位 0—2：为一个 11 位寄存器的高 2 位。其值定义窗口的水平结束。这个值由 VGA 像素时钟检测并由 VGA 水平同步的下降沿选取。

3—7：保留(0)

显示窗口，Y 结束低字节寄存器(R47)：读/写，索引：47H

位 0—7：为一个 10 位寄存器的低 8 位。其值定义窗口的垂直结束。这个值由 VGA 行检测并由 VGA 垂直同步的下降沿选取。

显示窗口，Y 结束高字节寄存器(R48)：读/写，索引：48H

位 0—1：为一个 10 位寄存器的高 2 位。其值定义窗口的垂直结束。这个值由 VGA 行检测并由 VGA 垂直同步的下降沿选取。

2—7：保留(0)

X 扫调，低寄存器(R49)：读/写，索引：49H

位 0—7：为一个 9 位寄存器的低 8 位。其值定义显示缓冲列地址乘 2，地址在 VRAM 的数据传送时被装入。对 4:1:1 编码，这个寄存器的 0 位应置为 0。

Y 扫调，低寄存器(R4A)：读/写，索引：4AH

位 0—7：为一个 9 位偏移寄存器的低 8 位。其值定义显示缓冲行。装入缓冲行用于第一现行显示行。

X，Y 扫调，高寄存器(R4B)：读/写，索引：4BH

位 0：列偏移的高位。(参看 R4A)

1—3：保密(0)

4：行偏移的高位。(参看 R4A)

5—7：保留(0)

移位时钟起始寄存器(R4C)：读/写，索引：4CH

位 0—6：这 7 位定义相对于 VGA 水平结束沿的显示消隐。

7：保密(0)

同步极性寄存器(R4D)：读/写，索引：4DH

位 0—1：水平放大，00:无放大，01:2X，10:4X，11:8X

2—3：垂直放大，00:无放大，01:2X，10:4X，11:8X

4：VGA 水平同步极性[POLARITY]，0:VGA 水平同步有效低，1:VGA 水平有效高。

5：VGA 垂直同步极性[POLARITY]，0:VGA 垂直同步有效低，1:VGA 垂直有效高。

6—7：保留(0)

彩色比较寄存器(R4E)：读/写，索引：4EH

位 0—7：这些位定义彩色匹配发生时 VGA 所必须具有的值。0:VGA 数据必须为 0，1：VGA 数据必须为 1。

彩色屏蔽寄存器(R4F)：读/写，索引：4FH

位 0—7：这些位定义与彩色值必须必须相符合的位置位。0:此位在 VGA 数据必须与彩色值相符合，1:此位在 VGA 数据中不用考虑。

显示窗口隔行扫描控制：读/写，索引：50H

位 0：使能隔行扫描显示。0 显示窗口为非隔行扫描，1 显示窗口隔行扫描。

1：选择外部场。0 显示窗口使用内部产生的场信号，1 为显示窗口场信号选择 VFLD 输入。

2：反转场。0 不修改场信号极性，1 修改显示窗口信号极性。

3：复制场。0 不复制，1 依位 4 复制奇或偶场。

4：奇/偶复制。0 如果位 3 已设置则复制偶场，1 如果位 3 已设置则复制奇场。

5—7：保留(0)

注：只有当位 0 为 1 时，位 1—2 才有效。如果位 1 是 0，则内部逻辑由采集在水平同步信号 VHSYNC 的前沿 4 个 PCLK 以后的垂直同步信号(HSYNC)电平来决定奇偶场。若 HSYNC 电平为低，则偶场，若电平为高则奇场。

芯片文本/使能寄存器(RFFH)：读/写，索引：FFH

位 0：PC 视频整体使能。要访别的任何一个 PC 视频寄存器此位必须设置。0 除了索引及整体使能寄存器外所有寄存器禁止，索引及整体寄存器为只写。1 所有寄存器包括索引及整体寄存器无论读和写都使能，但此位不可读。

1：使能内存。此位不可读。

2：IOWR/延时。此位不可读。0:IOWR/输入在芯片内部延时 2 XCLK 周期。1:IOWR/不延时。

3：保留(0)

4—7：这些位包含有 PC 视频的版本数。其值开始为 0，每种新芯片增加一。

3.3 视频模拟输入接口 TDA8708A

特征：

8 位分辨。

取样速率高达 32 兆赫。

二进或二的补码三态 TTL 输出。内部参考电压调节。

典型功耗 335 毫瓦。

视频输入可以三路选一。

CVBS(复合视频信号)钳位和自动增益控制(AGC)。

不须取样保持电路。

视频模数转换幅度为1.0伏(TDA8708 0.5伏)。

应用范围：

视频信号译码。

电视编码和解码。

数字图像处理。

俘获。

3.3.1 概述

视频信号处理的模拟输入接口。它包括一个带有钳位和增益控制的视频放大级，一个带取样速率为32兆赫8位模数转换和输入选择器。28脚双列直插封装。它的内部结构如图3.2所示。

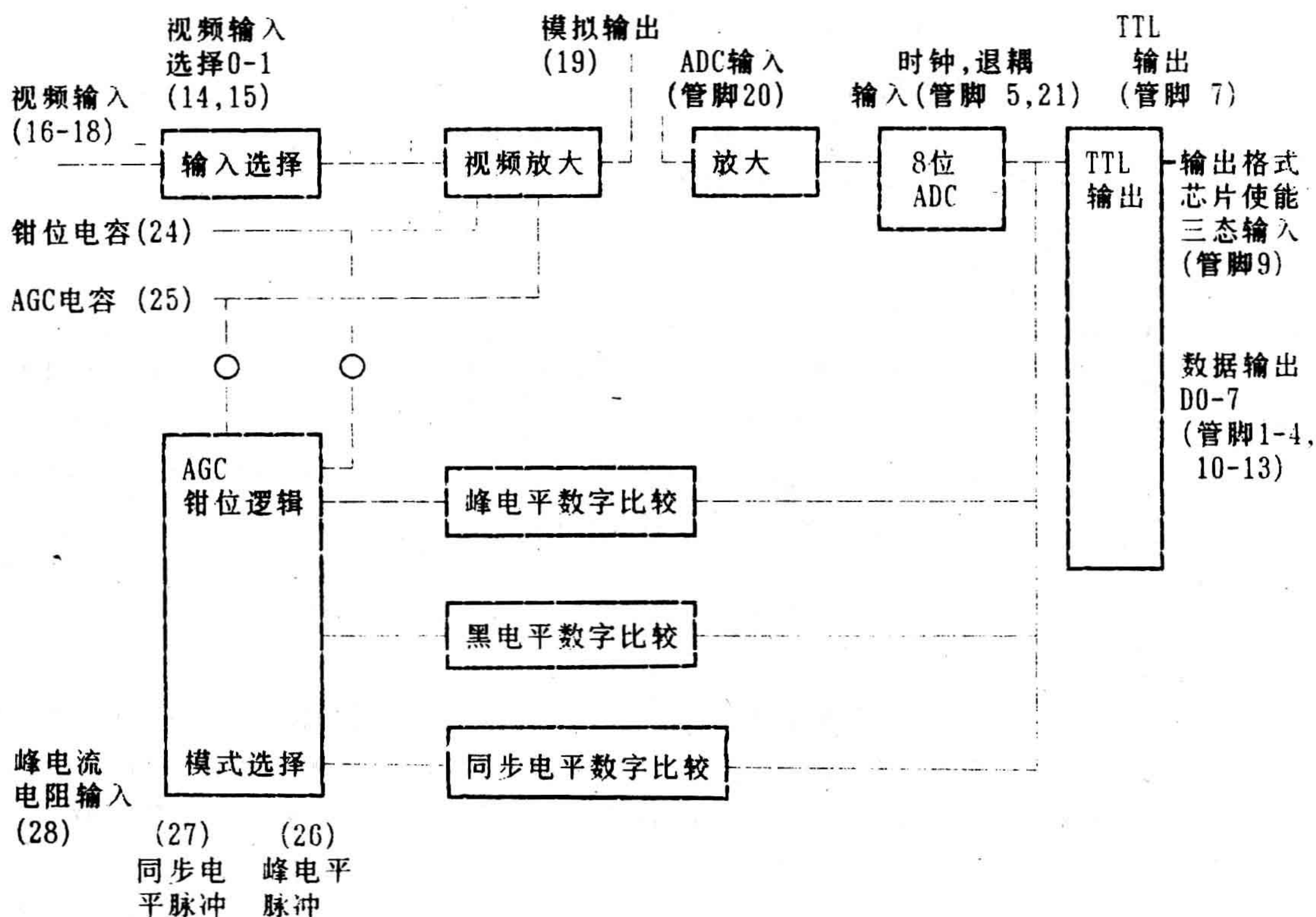


图3.2 TDAD8708A方框图

3.3.2 管脚名及功能

共有 28 个管脚，双列直插式。以下就其功能分类介绍：

管脚 1—4 (D7—4)：数据输出(高位)。

5 (CLK)：时钟输入。

6 (Vccd)：数字正供给电压(5V)

7 (Vcco)：TTL 输出正供给电压(5V)

8 (DGND)：数字地

9 (OF)：输出格式/芯片使能(3 态输入)

10—13 (D3—0)：数据输出(低位)

14—15 (I0—1)：视频输入选择 0 位和 1 位。

16—18 (VIN0—2)：视频输入 0—2

19 (ANOUT)：模拟电压输出。

20 (ADIN)：模数转换输入。

21 (DEC)：去偶输入。

22 (Vcca)：模拟正供给电压(+5V)

23 (AGND)：模拟地。

24 (CLAMP)：钳位电容联结。

25 (AGC)：AGC 电容联结。

26 (GATEB)：黑电平同步脉冲。

27 (GATEA)：同步电平的同步脉冲。

28 (RPEAK)：峰电平电流电阻输入。

3.3.3 功能描述

当视频信号弱，也就是说自动增益放大器的增益还没达到最佳值时，TDA8708A 工作于模式 1。它保证了译码电路迅速获得同步脉冲。当管脚 GATE A 和 GATE B 的输入变得可以区分时(GATE A 和 GATE B 分别是在同步周期和它的后沿的脉冲)TDA8708A 自动开关到模式 2 工作。在模式 1 时，自动增益放大器会进行粗调，同步电平对应于数字输出 0，峰电平对应于数字输出 255。模式 2 时模数转换的数字输出和内部的数字参考相比较。比较结果用来控制联到 AGC(自动增益控制)管脚的电容的充放电。电容上的电压控制视频放大器的增益。这是增益控制回路。

GATE A 输入的脉充正沿激活同步电平比较。这表明复合视频信号是用作为振幅参考。同步脉充的底被调节成数字输出的 0 电平。它的黑电平对应于 64。峰白控制回路一直处于工作状态，如果视频信号将要超过数字码 240，则增益就减小以避免模数转换的可能过载。

用标定的视频信号将不会超过数字码 213，这时峰白控制回路不工作。

钳位电平控制所用技术和增益控制的相同。GATE B 脉充正沿黑电平数字比较被激活。钳位电容被充放电来调节输出码到 64。

3.4 SAA9051 数字多制式电视译码器

特征：

在 13.5 兆赫的取样频率下，适用于各种传输制式；具有以内存为基础的特征。分开的色度和亮度输入(Y/C)。

标准应用复合视频信号输入。

SECAM(顺序同时制)应用中 CVBS 直通。

各种电视制式(PAL,NTSC,SECAM,B/W)下亮度信号处理。

各种电视制式下水平和垂直同步检测。

各种正交调幅彩色载波信号的色度信号处理。

只需一片晶振。

由 I2C 总线控制。

用户可程序孔径矫正(水平峰)。

与内存为基础的特征兼容(行锁相时钟)。

用色度梳状(comb)滤波器(NTSC)减少串色。

宽量程色调(Hue)控制。

3.4.1 概述

SAA9051 数字多制式译码器(S—DMSD)对所有制式的正交调制彩色电视信号解调和译码并且对具有 CVBS 或 Y/C 输入信号的各种电视制式的亮度信号处理。

它的方框图如图 3.3 所示。

3.4.2 管脚名及功能

共有 4 * 17 个管脚，管脚分布如图 3.4 所示。以下就其功能分类介绍：

管脚 1,5,27,28,35,37—39, 54,59—63,67 (no)：	没接
2 (TEST)：	检查输入(有效高)，当 HIGH(高)使能扫描检查模式，总是接地。
3 (RES/)：	复位输入(低有效)，导致 I2C 总线控制寄存器 1—3 和内部阶段在复位相时被复位。RES/最小的 LOW(低)长度为 120 个 LL3 时钟周期。
4 (LL3)：	13.5MHz 线锁系统时钟。
6—13 (IO0 低位—IO7 高位)：	双向数据路经。分开亮度和色度输入格式(Y/C)的色度输入或对 SECAM 译码器 SAA9056 的 CVBS 输出。
14—17,20—23 (CVBS0 低位—CVBS7 高位)：	数字化复合视频消隐和同步信号；包含亮度，色度和所有同步信息，或亮度，消隐和在分离亮度与色度(Y/C)输入时的同步信号。
18,52 (Vdd)：	正电源(+5V)

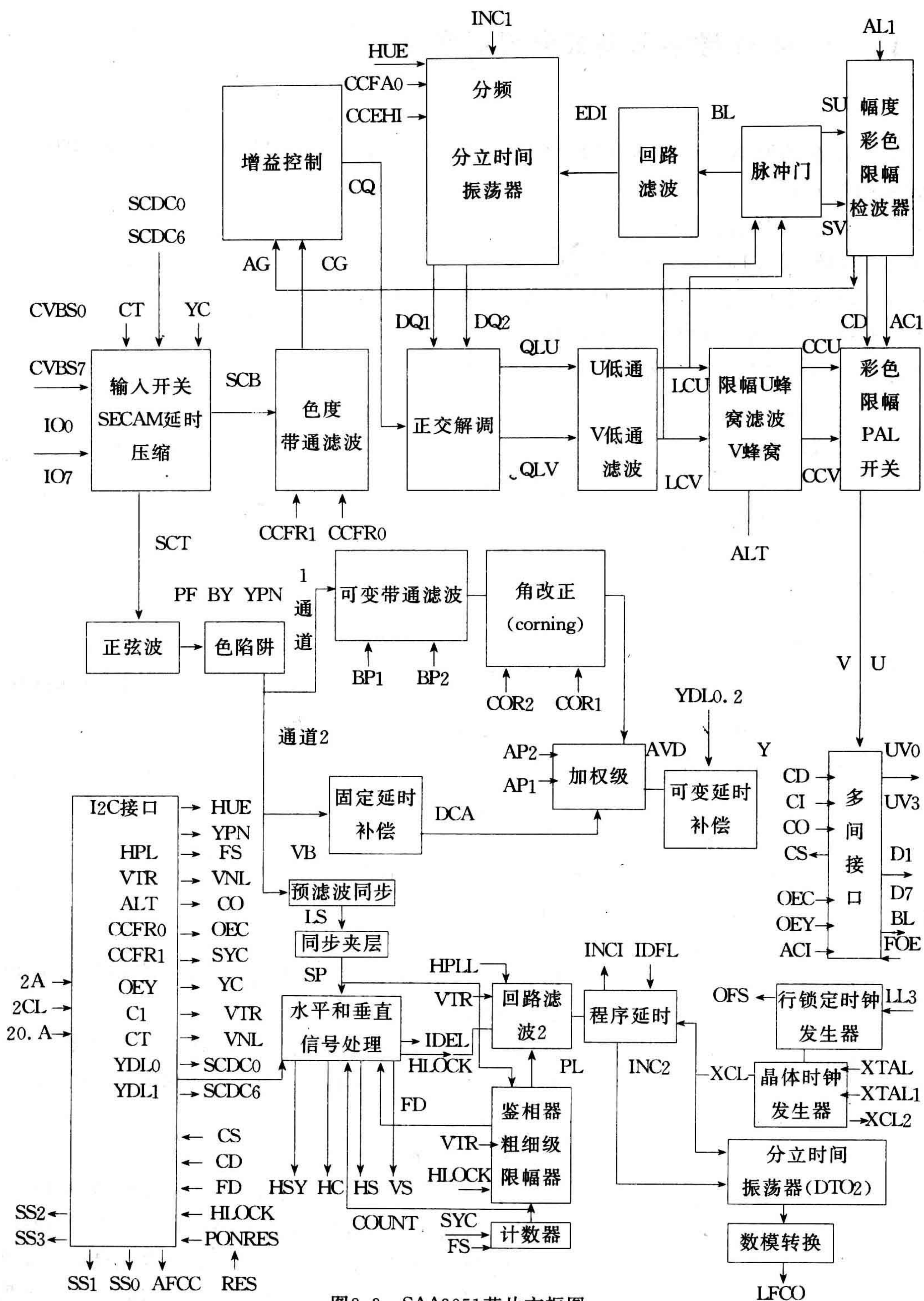


图3.3　SAA9051芯片方框图

19,51 (Vss):　　　　　　　地

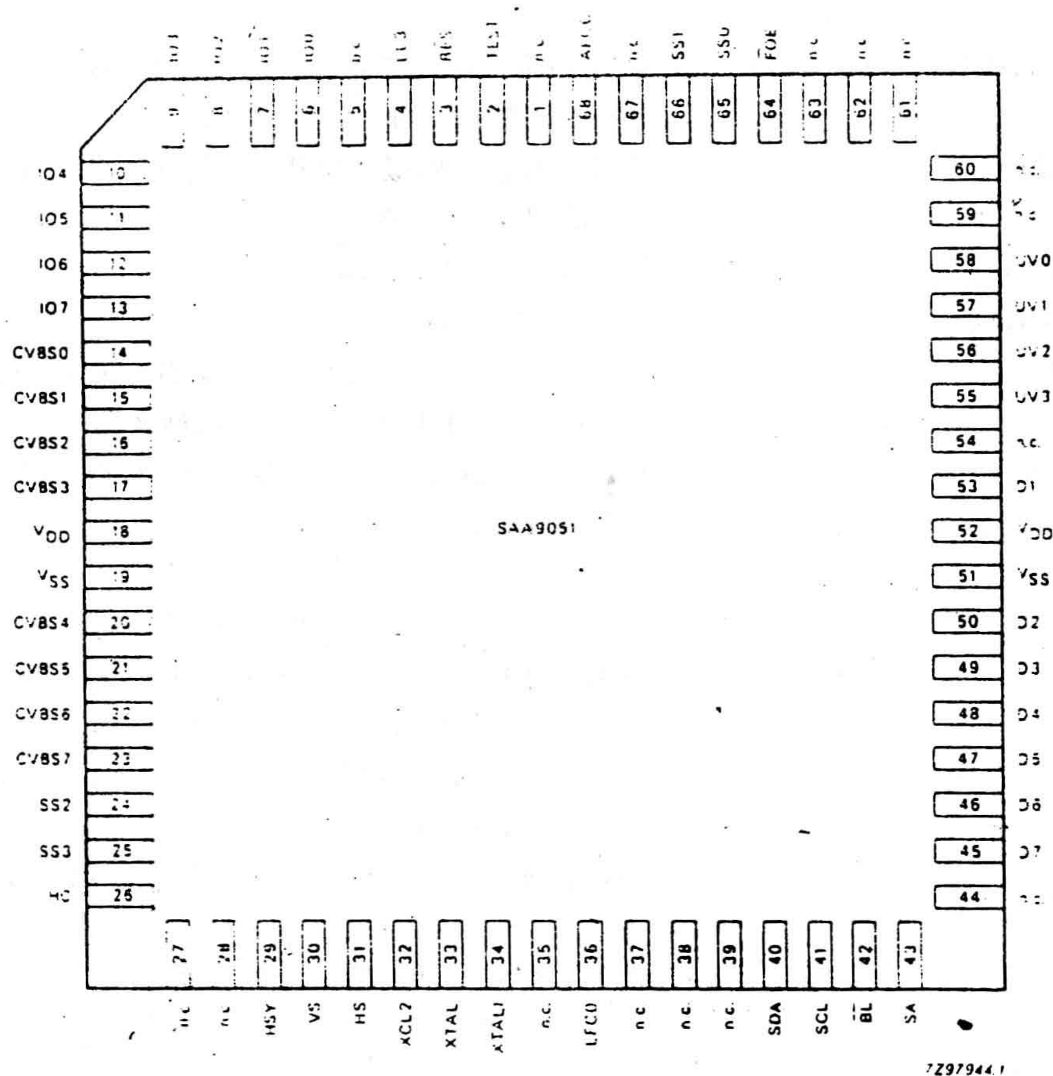


图 3.4 SAA9051 管脚分布图

24—25 (SS2—3)： 源选择输出信号；I2C 总线控制；TTL 兼容开关。

26 (HC)： 可程序水平输出脉冲；当用于联结输入线路时标志模数转换之前的黑电平位置。开始和结束时间是可程序的，在 I2C 总线上从－9.4μs 到＋9.4μs 步长为 74ns。

29 (HSY)： 可程序水平输出脉冲；当用于联结输入线路时标志模数转换之前的同步脉冲位置。开始和结束时间是可程序的，在 I2C 总线上从－14.2μs 到＋4.7μs 步长为 74ns。

30 (VS)： 垂直同步输出；标志 50/60Hz 场频图画的垂直位置。

31 (HS)： 水平同步脉冲输出(持续 64LL3 个时钟周器)。HS 是可程序的，在 I2C 总线上从－32μs 到＋32μs 步长为 300ns。

32 (XCL2)： 时钟输出，半晶振时钟频率(12.288MHz)。

33 (XTAL)：	晶振输入/转换放大输出；从外部晶振输入到内部时钟发生器。或转换放大输出到一个外部晶体(24.576)。
34 (XTALI)：	从外部晶体输入到内部转换放大器(24。576)。如果使用外部振荡连于地。
36 (LFCO)：	线频率控制. 模拟输出代表一个有 4 位分辨率线频率(6.75MHz)的多路。
40 (SDA)：	I2C 总线串行输入/输出。
41 (SCL)：	I2C 总线时钟输入。
42 (BL)：	消隐信号输出(低有效)；标志有效视频和线消隐时间。BL 也同步数据多路及非数据多路。
43 (SA)：	I2C 总线选择地址；选择适当 I2C 从地址的输入。
45－50,53 (D7 高位－D1 低位)：	亮度数据输出。
55－58 (UV3－0)：	多路 PAL 或 NTSC 彩色差别信号输出或从 SECAM 译码器来的 SECAM CS 输入信号。输出数据格式为 2 的补码。多路同步于 BL/的上升沿。
64 (FOE/)：	快速输出使能信号(低有效)。设置 D1－7 和 UV0－3 输出为高阻 z 状态。
65－66 (SS0－SS1)：	源选择输出信号，设置经 I2C 总线。用于控制输入开关。
68 (AFCC)：	由 I2C 总线激活的附加电路控制输出。

3.4.3 功能描述

对所有正交调制彩色电视制式(PAL B,G,H,I,M,N,NTSC 4.43MHz 和 NTSC M)解调和译码还对部分制式(PAL,NTSC 和 SECAM)的色度和同步信号处理。所有这些控制功能，无论用户或工厂调节，都由 I2C 总线存取从而增强了数字电视观念的适应性。

操作基于行锁定 13.5MHz 采样频率，这样使系统完全适用于任何行频。对所有电视制式只需一片晶片。它应和 SAA9057A 时钟发生电路(CGC)配套使用，否则设计者应保证电源失效时应给 S－DMSD 一复位脉冲。

3.4.4 Y/C 处理

色度信号经 IO 口(IO0－7)输入并且传输经过输入开关/SECAM 延时补偿电路(多路的)到彩色带通滤波器，参看色度通道。其余的 Y 信号和同步脉冲信号由 CVBS0－7 输入，经输入开关/SECAM 延时补偿电路到亮度预滤波器。

3.4.5 复合电视信号处理

复合视频信号由色陷阱和带通滤波分成亮度(VBS)和色度(CG)两分量。这些电路可由制式确认信号(CCFR0,1/YPN)按照检测的彩色载频是 3.58 或 4.43MHz 来开关。在接收 SE-

CAM 信号时，信号经 IO 口(IO0－7)送到 SECAM 译码器(SAA9056)。位 CT 使两部分之间三态缓存使能。

3.4.6 亮度通道

色陷阱之后，亮度分为三通道：

通道 1 信号：送到可程序带通滤波器(中心频率可由 BP1 和 BP2 位来编程)滤去亮度的高频分量。广播(BC)信号送到核级(角校正)滤去低振幅的噪声(噪声幅度滤调的数量可由 COR1 和 COR2 位编程)。高频分量送到加权级，参看 1 和 2 信号复合通道。通道 2 信号：送到固定延时补偿级进行固定延时和黑电平调节。DCA 信号传送到加权级。

1 和 2 信号复合通道：通道 1 的高频信号 2 加权到通道 2 的 DCA 信号。复合信号调节到一定幅度只占 7 位。AVD 信号传到可变延时补偿级对中频信号群延时补偿。延时量可由 YDL0－2 三位编程(从－4 到＋3 LL3 时钟周期。Y 信号由分时多路接口经 D0－7 输出。通道 3 信号：送到预滤波同步级，参看同步通道。

3.4.7 色度通道

色度 CG 信号从色度带通级传送到增益控制电路。增益控制级保证色度信号有恒定幅度的脉冲串。GQ 信号被传送到正交解调器，在那里正交调解的色度 GQ 信号解调到彩色差别信号。QLU 和 QLV 信号传送到一个低通滤波器。LCU 和 LCV 信号传送到一个限制器和梳状滤波级。对 NTSC 梳状滤波级分离剩余的垂直相关的亮度分量(对 PAL，信号为相角校正)。CCU 和 CCV 信号传送到彩色删除和 PAL 开关级。在此阶段除去与选择标准不一致的信号。在 PAL 模式中，此阶段恢复 V 信号的矫正相。然后，信号传递到时间多路接口经过 UV0－3 输出。

3.4.8 彩色载频再生

彩色载频再生由锁相回路(PLL)实现。锁相环路包括正交解调，低通滤波，短促门，环路滤波 1 和分离/离散时间振荡器(DTO1)。DTO1 由标准的统一信号 CCFR0－CCFR1 及影响色度信号解调相的 Hue 信号控制。

3.4.9 同步通道

在同步电路中前置滤波器同步使同步脉冲斜率正常化。同步限幅器提供检测同步脉冲到水平与垂直过程和相检测器阶段。水平与垂直过程：水平与垂直过程包括再产生水平同步(HS)的部分 PLL 电路和一个用于垂直(VS)同步检测的适合的滤波器。水平与垂直过程也产生控制哑功能的同时信号和统一正常每幅 525 或 625 行的统一信号。相检测器：相检测器接收 SP 信号和部分 PLL，控制行锁时钟(PL)的产生。依赖时间常数信号具有变带宽的回路滤波器 2 产生两个不同延迟的增量信号(INC1 和 INC2)。INC2 经过增量延迟信号(IDEL)是可程序的。在 DTO1，INC1 校正再产生副载频，INC2 实现 DTO2 的相增加。晶体时钟产生器供给稳定的 24.576MHz 时钟输入到 DTO2，DTO2 又供给有 432 或 429 倍线频率的数字控制信号的 4 位 DAC。模拟输出 LFCO 从 DAC 传送到 SAA9057A(CGC)。

3.4.10 输出接口

信号 OEY,OEC,CO,CI,和 CD 控制输出接口。除了在 S—DMSD 中被检测的 CD 信号外,其它信号经过 I2C 总线接受。电源开复位使这些信号为零。

表 1 垂直噪声限幅器(VNL 信号):输出 0,VNL 旁路;输出 1,VNL 激活。

表 2 CO,CI,和 CD 信号

CO	CI	CD	输出	输出状态
0	×	×	UV0—3	彩色 OFF(0)
1	0	0	UV0—3	彩色 OFF(由 CD 控制)
1	0	1	UV0—3	彩色 ON(由 CD 控制)
1	1	X	UV0—3	强行彩色 ON

这里:×不用管

表 3 OEC,OEY,FOE/,BL/,D1—7 和 UV0—3 信号

OEC	OEY	FOE/	BL/,S,HS	D1—7	UV0—3	说明
0	0	×	HIZS	HIZS	HIZS	电源复位(ON)后的状态
1	1	1	激活	HIZS	HIZS	
1	1	0	激活	激活	激活	
0	1	1	激活	HIZS	HIZS	
0	1	0	激活	激活	激活	

这里:× 不用管

HIZS=高阻 Z 态。

注:不允许超出以上的结合。

CS 信号

CS 信号在水平消隐时期从数字 SECAM 译码器(DSD)传送,经 UV2 输入接收。

3.4.11 I2C 总线接口

下列控制信号经 I2C 总线接口接收:

标准统一信号(CCFR0,CCFR1,ALT,FS,YPN)

录像机/电视时间常数(VTR)

色调控制(Hue)

水平信号延迟编程(HS,HC,HSY)

增量延迟(IDEL)

亮度孔径矫正控制(BY,PF,BP1,BP2,COR2,COR1,AP2,AP1)

亮度延迟补偿(YDL0,YDL1,YDL2)

固定时钟产生命令(HPLL)

内部彩色 ON/OFF(CO)

内部强行彩色 ON,仅检查目的(CI)。

垂直噪音限制(VNL)激活/旁路

亮度和同步使能(OEY)

色度输出使能(OEC)
开关信号(源选择信号 SS0,SS1,SS2,SS3)
电路控制附加输出(AFCC)
色度源选择 CVBS/色度输入/输出(CT/YC)
SECAM 色度延迟补偿(SCDC0,SCDC1,SCDC2,SCDC3,SCDC4,SCDC5,SCDC6)
水平同步(HSY)和钳位(HC)脉冲禁止(SYC)
经 I2C 总线从 S—DMSD 的信号传送为：
标准统一信号(FD,CS)
彩色删除状态信号(CD)
一致信息(HLOCK)
S—DMSD 开电源复位(PONRES)

表 4　从地址

从接收器地址									
SA	A6	A5	A4	A3	A2	A1	A0	.	说明
0	1	0	0	0	1	0	1	0	(8AHEX)
1	1	0	0	0	1	1	1	0	(8EHEX)

这里:. ＝逻辑 0 为接收模式
. ＝逻辑 1 为发送模式

表 5　子地址字节和数据字节格式

寄存器功能	子地址	数据:D7	D6	D5	D4	D3	D2	D1	D0
增量延迟(IDEL)	00	A07	A06	A05	A04	A03	A02	A01	A00
HSY 开始时间	01	A17	A16	A15	A14	A13	A12	A11	A10
HSY 结束时间	02	A27	A26	A25	A24	A23	A22	A21	A20
HC 开始时间	03	A37	A36	A35	A34	A33	A32	A31	A30
HC 开始时间	04	A47	A46	A45	A44	A43	A42	A41	A40
HS 开始时间(PHI1 后)	05	A57	A56	A55	A54	A53	A52	A51	A50
水平峰	06	BY	PF	BP2	BP1	COR2	COR1	AP2	AP1
色调控制	07	A77	A76	A75	A74	A73	A72	A71	A70
控制 1	08	HPLL	FS	VTR	CO	ALT	YPN	CCFR1	CCFR0
控制 2	09	VNL	OEY	OEC	×	CI	AFCC	SS1	SS0
控制 3	0A	SYC	CT	YC	SS3	SS2	YDL2	YDL1	YDL0
SECAM 延迟补偿	0B	×	SCDC6	SCDC5	SCDC4	SCDC3	SCDC2	SCDC1	SCDC0
保留	0C,0F	×	×	×	×	×	×	×	×

这里：×不用管

表 6　增量延迟控制 IDEL

十进制乘数	延迟时间(步长＝2/13.5MHz＝148ns)	控制位 A07	A06	A05	A04	A03	A02	A01	A00
—1 到	—148ns(最小值)	1	1	1	1	1	1	1	1
—110	—16.3μs(外部有效范围)	1	0	0	1	0	0	1	0
—111 到	—16.44μs	1	0	0	1	0	0	0	1

−214	−31.7μs(最大值,若 FS=1)	0	0	1	0	1	0	1	0
−215	−31.85μs(外部中心计数,若 FS=1)	0	0	1	0	1	0	0	1
−216	−32μs(最大值,若 FS=0)	0	0	1	0	1	0	0	0
−217 到	−32.148μs(外部中心计数,若 FS=0)	0	0	1	0	0	1	1	1
−256	−37.9μs(外部中心计数)	0	0	0	0	0	0	0	0

子地址 01

表 7　水平同步 HSY 开始时间

十进制乘数	延迟时间(步长=1/13.5MHz=74ns)	控制位 A17	A16	A15	A14	A13	A12	A11	A10
+191 到	−14.2μs(最大负值)	1	0	1	1	1	1	1	1
+1	−0.074μs	0	0	0	0	0	0	0	1
0	0μs 参考点	0	0	0	0	0	0	0	0
−1 到	+0.074μs	1	1	1	1	1	1	1	1
−64	+4.7μs(最大正值)	1	1	0	0	0	0	0	0

子地址 02

表 8　水平同步 HSY 结束时间

十进制乘数	延迟时间(步长=1/13.5MHz=74ns)	控制位 A27	A26	A25	A24	A23	A22	A21	A20
+191 到	−14.2μs(最大负值)	1	0	1	1	1	1	1	1
+1	−0.074μs	0	0	0	0	0	0	0	1
0	0μs 参考点	0	0	0	0	0	0	0	0
−1 到	+0.074μs	1	1	1	1	1	1	1	1
−64	+4.7μs(最大正值)	1	1	0	0	0	0	0	0

子地址 03

表 9　水平嵌位 HC 开始时间

十进制乘数	延迟时间(步长=1/13.5MHz=74ns)	控制位 A37	A36	A35	A34	A33	A32	A31	A30
+127 到	−9.4μs(最大负值)	0	1	1	1	1	1	1	1
+1	−0.074μs	0	0	0	0	0	0	0	1

0	0μs 参考点	0	0	0	0	0	0	0	0
−1 到	+0.074μs	1	1	1	1	1	1	1	1
−128	+9.5μs(最大正值)	1	0	0	0	0	0	0	0

子地址 04

表 10 水平嵌位 HC 结束时间

十进制乘数	延迟时间(步长=1/13.5MHz=74ns)	控制位 A47	A46	A45	A44	A43	A42	A41	A40
+127 到	−9.4μs(最大负值)	0	1	1	1	1	1	1	1
+1	−0.074μs	0	0	0	0	0	0	0	1
0	0μs 参考点	0	0	0	0	0	0	0	0
−1 到	+0.074μs	1	1	1	1	1	1	1	1
−128	+9.5μs(最大正值)	1	0	0	0	0	0	0	0

子地址 05

表 11 PHI1 后水平同步 HS 开始时间，50Hz，625 行(FS=0)

十进制乘数	延迟时间(步长=4/13.5MHz=296ns)	控制位 A57	A56	A55	A54	A53	A52	A51	A50
+127 到	禁止，外部有效中心计数范围	0	1	1	1	1	1	1	1
+109	禁止，外部有效中心计数范围	0	1	1	0	1	1	0	1
+108 到	−32μs(最大负值)	0	1	1	0	1	1	0	0
+1	−0.296μs	0	0	0	0	0	0	0	1
0	0μs 参考点	0	0	0	0	0	0	0	0
−1 到	+0.296μs	1	1	1	1	1	1	1	1
−107	+31.7μs(最大正值)	1	0	0	1	0	1	0	1
−108 到	禁止，外部有效中心计数范围	1	0	0	1	0	1	0	0
−128	禁止，外部有效中心计数范围	1	0	0	0	0	0	0	0

表 12 PHI1 后水平同步 HS 开始时间，50Hz，625 行(FS=1)

十进制乘数	延迟时间(步长=4/13.5MHz=296ns)	控制位 A57	A56	A55	A54	A53	A52	A51	A50

+127 到	禁止,外部有效中心计数范围	0	1	1	1	1	1	1	1
+107	禁止,外部有效中心计数范围	0	1	1	0	1	0	1	1
+106 到	−31.8μs(最大负值)	0	1	1	0	1	0	1	0
+1	−0.294μs	0	0	0	0	0	0	0	1
0	0μs 参考点	0	0	0	0	0	0	0	0
−1 到	+0.294μs	1	1	1	1	1	1	1	1
−107	+31.5μs(最大正值)	1	0	0	1	0	1	0	1
−108 到	禁止,外部有效中心计数范围	1	0	0	1	0	1	0	0
−128	禁止,外部有效中心计数范围	1	0	0	0	0	0	0	0

3.4.12 IDEL,HSY,HC 和 HS 编程

在 I2C 总线上，用数据字把变量 IDEL,HSY,HC 和 HS 编程。在下面的例子中其值随着时间的增加而减小。

IDEL： (参看图 3.4.1)IDEL 数据字用于环路滤波器 2，正交解调器和内部/外部(系统)信号通道间的补偿。内部延迟为 INC1 通过环路滤波器 2,·分离器和 DTO1 所需时间。该延迟矫正负载频与线载频的关系外部通道是下列时间延迟的结果(LL3 时钟周器)：

tDEL： 可程序延时

ta： DTO2 和 DAC 的处理时间

tb： 色度带 Ⅱ 和增益控制阶段延时

tCGC： 时钟产生器电路延时

tADC： 模数转换器延时

tINP： 输入开关延时

HSY： HSY 编程范围(START/STOP)为+191 到−64(LL3 时钟周期)

HC： HC 编程范围(START/STOP)为+127 到−128(LL3 时钟周期)(参看图 3.4.2)

HS： 移 HS 脉冲到消隐脉冲 BL/的中心这样计算：对 NTSC 和 PAL 模式，HS 相对于 0 点位置为 4LL3 时钟周期。HS 周期长度是 64LL3 时钟周期。

3.4.13 S−DMSD 亮度通道编程

没有色度的 VBS 信号或 CVBS 进入最大增益为 9.5 dB 的高通传递函数的预滤波级。控制位 PF 把滤波器开关到旁路方式。下一级是色陷阱，它的零点可以由 YPN 控制位编程为 4.43 兆赫(PAL)或 3.58 兆赫(NTSC)。BY 位激活 S−DMSD 的 Y/G 方式旁路功能。色陷阱输出信号然后分成三路，参看亮度通道一节。

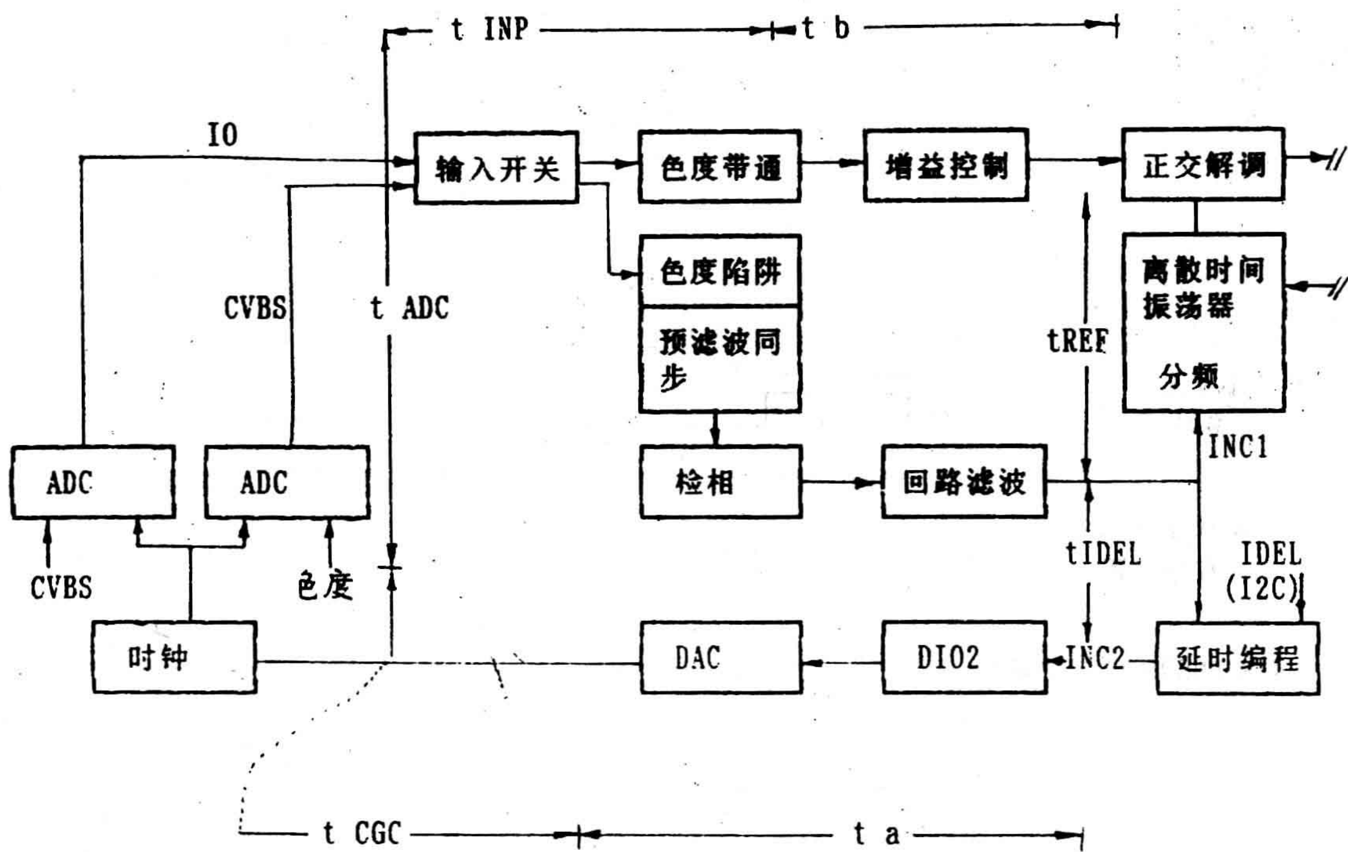


图 3.4.1 增量延时控制 IDEL 作延时补偿

子地址 06

表 13 色度陷阱选择(BY 开关色度陷阱到旁路模式,YPN 选择槽口频率)

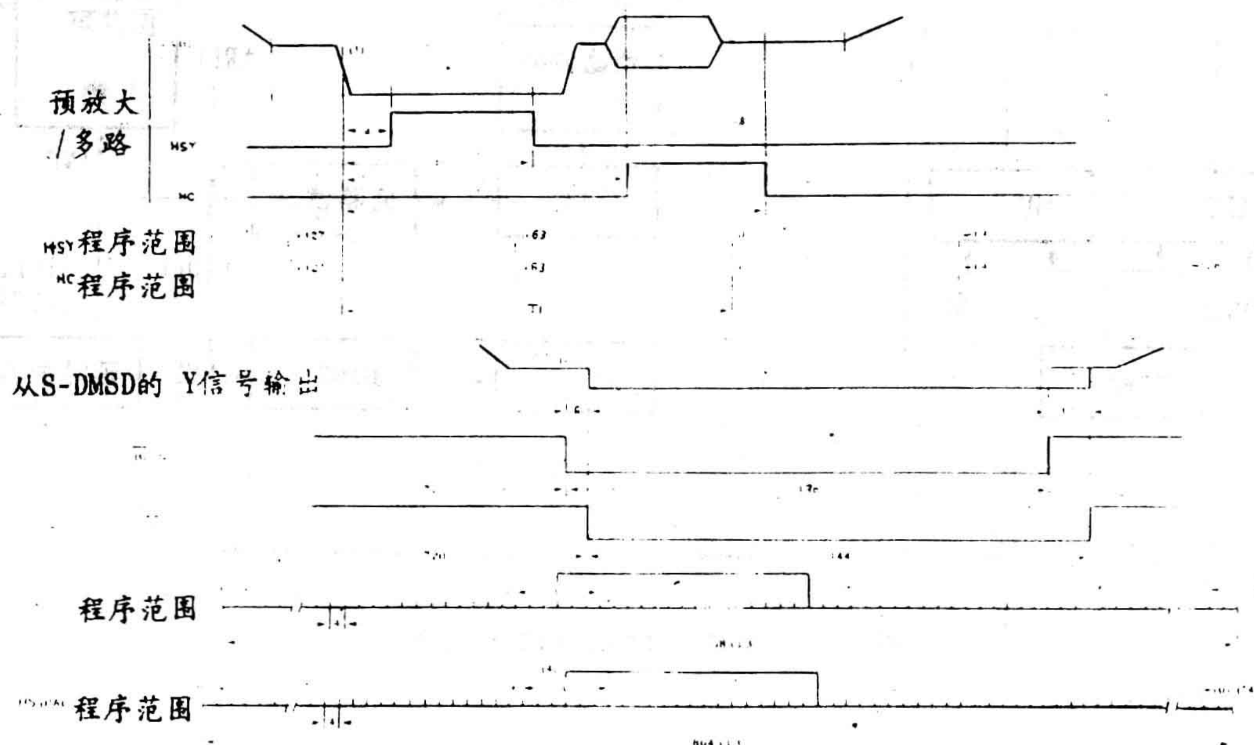
色度陷阱	控制位:BY(SA06,D7)	YPN(SA08,D2)
PAL(4.43MHz)	0	0
NTSC(3.58MHz)	0	1
旁路	1	×

14 断开亮度前置滤波

前置滤波	控制位:PF(SA06,D6)
ON	0
OFF	1

表 15 带通控制(BP1 和 BP2 控制带通滤波器的中心频率)

带通类型(中心频率)	控制位:BP2(SA06,D5)	BP1(SA06,D4)
类型 1(4.1MHz)	0	0
类型 2(3.8MHz)	0	1
类型 3(2.6MHz)	1	0
类型 4(2.9MHz)	1	1



HSY 和 HC是参照于模拟输入信号(1).BL 和 HS 输出参照于S-DMSD 输出(2).波形时序用
LL3 时钟周期的 n倍表示.HSY,BL和输入到S-DMSD的CVBS n=1,HS n=4.
在子地址SA06=C0输入到输出的数据延时 T1
SA0B=00:63　　SA0B=3C:　3

图 3.4.2　信号校正

表 16　核临界电平(COR1 和 COR2 控制低放大和高频信号分量的抑制)

临界	控制位：COR2(SA06,D3)	COR1(SA06,D2)
核关	0	0
核开(12 位的 4 位)	0	1
核开(12 位的 5 位)	1	0
核开(12 位的 6 位)	1	1

注：临界与 12 位带通滤波器的字宽有关

表 17　正交矫正因子(AP1 和 AP2 选择亮度分量高频 HF 的加权因子 K)

加权因子 K	控制位：AP2(SA06,D1)	AP1(SA06,D0)
0	0	0

0.25	0	1
0.5	1	0
1	1	1

子地址 07

表 18　色调相

色调相	控制位：A77	A76	A75	A74	A73	A72	A71	A70
+178.6 到 0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	0	0	0
0 到 −180	0	0	0	0	0	0	0	0

子地址 08

表 19　水平时钟

功能	HPLL 控制位(SA08,D7)
水平时钟开,水平频率固定	1
水平时钟关	0

表 20　场频率选择

功能	控制位(SA08,D6)
60Hz,525 行模式	1
50Hz,625 行模式	0

表 21　VTR/TV 模式选择

功能	控制位 VTR(SA08,D5)
VTR 模式	1
TV 模式	0

表 22　彩色控制

功能	控制位 CO(SA08,D4)
彩色 ON	1
彩色 OFF	0

表 23　隔行/非隔行模式

功能	控制位 ALT(SA08,D3)
隔行模式(PAL)	1
非隔行模式(NTSC)	0

表 24　色度陷阱选择和振幅匹配

色度陷阱	控制位 YPN(SA08,D2)
3.58MHz	1
4.43MHz	0

表 25 彩色载频控制

彩色载频	控制位:CCFR1(SA08,D1)	CCFR(SA08,D0)
4,433,618.75Hz(PAL－B,G,H,1;NTSC 4.43)	0	0
3,575,611.49Hz(PAL－M)	0	1
3,582,056.25Hz(PAL－N)	1	0
3,579,545Hz(NTSC－M)	1	1

子地址 09

表 26 垂直噪声限制器

功能	控制位:VNL(SA09,D7)
VNL 激活	1
VNL 旁路	0

表 27 Y 输出使能

功能	控制位:OEY(SA09,D6)
输出 D0－7 和 BL/激活	1
输出 D0－7 和 BL/高阻 Z 态	0

表 28 色度输出使能

功能	控制位:OEC(SA09,D5)
输出 UV0－3 激活;CD 为 1,色度输出，为 0,则 0 信号	1
输出 UV0－3 高阻 Z 态	0

表 29 内部彩色强行 ON/OFF

功能	控制位:CI(SA09,D3)
ON,若 CO=1(CD=×)或 OFF,若 CO=0(CD=×)	1
OFF,若 CO=0(CD=×)或 CO=1,彩色由 CD 控制	0

表 30 电路控制附加输出

功能	控制位:AFCC
输出 AFCC=HIGH	1
输出 AFCC=LOW	0

表 31 源选择

功能	控制位:SS0－SS3
输出 SS0－3=HIGH	1
输出 SS0－3=LOW	0

表 31 源选择

控制位	管脚	从地址
AFCC	68	09,D2

SS3	25	0A,D4
SS2	24	0A,D3
SS1	66	09,D1
SS0	65	09,D0

子地址 0A

表 33　HSY 和 HC 脉冲的禁止

功能	控制位:SYC(SA0A,D7)
HSY 和 HC 脉冲禁止	1
HSY 和 HC 脉冲使能	0

表 34　色度输入/输出 3 态控制

功能	控制位:CT(SA0A,D6)
CVBS 输出激活	1
输出高阻	0

表 35　色度源选择

功能	控制位:YC(SA0A,D5)
Y/C 分开输入	1
CVBS 输入	0

表 36　亮度通道变量延迟补偿(YDL0—YDL2 控制亮度延迟为补偿不同于色度的延迟

延迟(N=)	YDL2	YDL1	YDL0
0	0	0	0
	0	0	1
2	0	1	0
3	0	1	1
—4	1	0	0
—3	1	0	1
—2	1	1	0
—1	1	1	1

子地址 0B

表 37　SECAM 色度延迟补偿

可程序延迟	控制位:SCDC6	SCDC5	SCDC4	SCDC3	SCDC2	SCDC1	SCDC0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1
2	0	0	0	0	0	1	0
4	0	0	0	0	1	0	0
8	0	0	0	1	0	0	0
16	0	0	1	0	0	0	0

32	0	1	0	0	0	0	0
63	0	1	1	1	1	1	1
64	1	1	1	0	0	0	0
65	1	1	1	0	0	0	1
79	1	1	1	1	1	1	1

从传输格式：

开始条件	从地址	来自从设备的确认	数据字节	是否确认	停止条件
S	1 0 0 0 1 SA 1 1	A	D7—D0	N/A	P

表 38 数据字节为

D7	D6	D5	D4	D3	D2	D1	D0
PONRES	HLOCK	1	FD	0	CD	CS	0

表 39 数据字节说明

PONRES：断电后电源供给复位(RES/)状态位。当断电或电压降低使 PAL/NTSC 译码器(SAA9051)中从数据受干扰时，该位为 1。PONRES 使寄存器 1 和 2 的所有数据位为 0。当成功的读从 PAL/NTSC 译码器中读到状态字节时，该位为 0。

HLOCK：水平频率锁的状态位。若水平频率未锁(没有有效传送)，该位为 1；若水平频率已锁(传送收到)，该位为 0。

FD：检测的场频率状态位。当收到信号为 60Hz 同步脉冲时该位为 1，当收到信号为 50Hz 同步脉冲时该位为 0。

CD：PAL/NTSC 彩色检测状态位。当 PAL/NTSC 彩色信号被检测到时，该位为 1；当 PAL/NTSC 彩色信号未被检测到时，该位为 0。

CS：SECAM 彩色检测状态位。当 SECAM 彩色信号被检测到时，该位为 1；当 SECAM 彩色信号未被检测到时，该位为 0。

3.4.14 S—DMSD 和 SAA9056 缺省系数集

缺省系数集为与 TDA8703 或 TDA8708 一起操作的模数转换器。色度 ADC 的三态输出由 SS3 开关控制。从地址为：S—DMSD；8A 或 8E SAA9056；8A 或 8E。

表 40 从地址(SAA9051 部分)

子地址	功能	短延迟	长延迟
00	增加延迟	5E	7E
01	HSY 开始	37	73
02	HSY 结束	07	43
03	HC 开始	F6	
04	HC 结束	C7	03
05	HS 开始	FF	FF
06	H—峰	02(62NTSC)	02(62NTSC)
07	HUE 控制	00	00

08	控制 1	38(77NTSC)	38(77NTSC)
09	控制 2	E3	E3(D3 SECAM)
0A	控制 3	58(28Y/C 模式)	58(28 Y/C 模式)
0B	SECAM 延迟	00	3C

表 41 从地址(SAA9056 部分)

子地址	功能	值
10	亮度延迟	C0—FF
11	BL/延迟	00
12	短促门开始	42
13	短促门结束	56
14	敏感性	20
15	滤波器	24
16	控制	04(02 激活)

表 42 S—DMSD 的操作模式

INPUT	CT	YC	SS3	CE	SCDC	IDEL	YPN	BY	FS	ALT	CCFR1	CCFR0
PAL B,G,H,I												
CVBS	1(0)	0	1(0)	0	B(A)	B(A)	0	0	0	1	0	0
Y/C	0	1	0	0	A	A	0(1)	1	0	1	0	0
PAL M												
CVBS	1(0)	0	1(0)	0	B(A)	B(A)	1	0	1	1	0	1
Y/C	0	1	0	0	A	A	1(0)	1	1	1	0	1
PAL N												
CVBS	1(0)	0	1(0)	0	B(A)	B(A)	0	0	0	1	1	0
Y/C	0	1	0	0	A	A	0(1)	1	0	1	1	0
SECAM												
CVBS	1	0(1)	1	1	B	B	0	0	0	0(1)	0(1)	0(1)
Y/C	0	1(0)	0	1	B	B	0(1)	1	0	0(1)	0(1)	0(1)
NTSC4.43MHz												
CVBS	1(0)	0	1(0)	0	B(A)	B(A)	0	0	0	0	0	0
Y/C	0	1	0	0	A	A	0(1)	1	1	0	0	0
NTSC M												
CVBS	1(0)	0	1(0)	0	B(A)	B(A)	1	0	1	0	1	1
Y/C	0	1	0	0	A	A	1(0)	1	1	0	1	1

3.5 数字电视系统时钟信号发生电路 SAA9057A

它由一个时钟输入经锁相回路产生四,三和二倍频并且有相移控制的时钟信号。适用于数字电视系统。

它的框图如下(图 3.5):

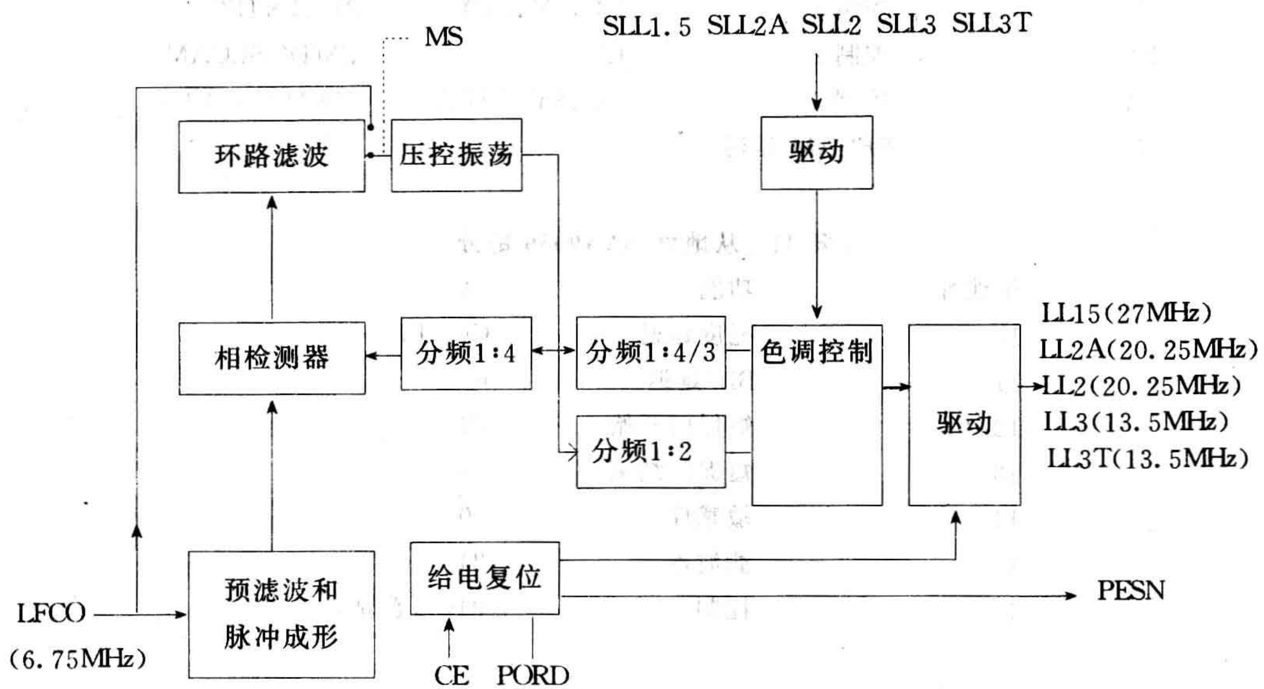


图3.5 SAA9057方框图

一般它和 74F244 高速缓冲器配合使用来保证推动能力和相移。参看下一章 GRANDVIDEO 电路图。对于 GRANDVIDEO，改用了 74F04 芯片。

3.6 视频控制和自动截止白电平控制 TDA4680

TDA4680 对具有色差接口的电视接收器实现视频控制功能。需要输入的信号是亮度和负色差 －(R－Y)和 －(B－Y)，2 或 3 级[SANDCASTLE]脉冲(用于时序控制)。RGB 输出信号对驱动视频输出级是有效的。该电路提供自动截止和白平衡控制。

特征：色差的电容偶合，有黑电平钳位的亮度和 RGB 输入。

经快开关或 I2C 总线两个完全被控制的模拟 RGB 输入使能。

经 I2C 总线饱和，对比和亮度控制在所有选择的信号上起作用。

为电视和嵌入信号的等黑电平。

自动截止的钳位，水平和垂直消隐时序和白平衡控制由 2 或 3 级沙堡[SANDCASTLE]脉冲控制。

补偿显像管泄漏电流的自动截止控制。

数字存储的自动白平衡控制或经 I2C 的简单白控制。

PAL，SECAM 和 NTSC/PAL－M 测量脉冲经 I2C 总线在可选择的消隐间隔出现在最后 3 或 4 行。

准备顺次扫描和经 I2C 总线 100Hz 应用。

在不变色的前提下为插入部分的 2 个开关开延迟。

平均束电流限制。

经 I2C 总线可调整的峰驱动限制。

经 I2C 总线为自动截止和白平衡控制的三个可调参数。

为驱动 RGB 输出级的发射流输出。

TDA4555 或 TDA4650 色调设置的 DAC。

框图见图 3.6：

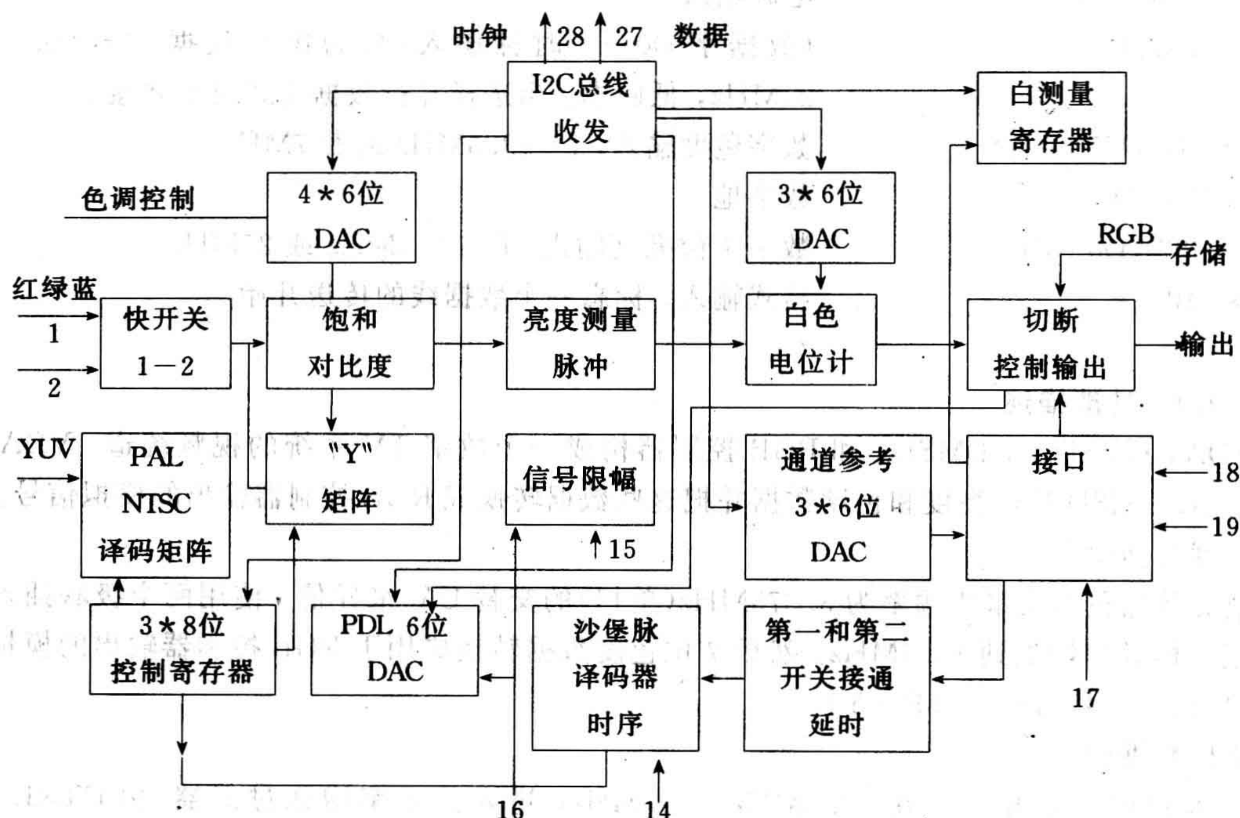


图3.6　TDA4680方框图

3.7　数模变换视频处理器 SAA9060

该芯片是一视频处理器，具有把数字亮度和色度数据为 RGB 控制器转换成为模拟信息的数模变换器 DACs（VDA）。

特征：单扫描或双扫描

　　并行数据输入

　　彩色差信号的 7 位 D/A 转换

　　亮度信号的 8 位 D/A 转换

此芯片为双列直插式，其管脚说明如下：

管脚：2，3	未接
1（Vsub）	基底管脚
4，5（CTRL1－0）	为分辨增强滤波器的控制输入
6（OS）	输入 D0－6，UV0－3，BLN 数据格式转换（从串行到并行）
7（Vssa）	模拟地
8（IBIAS）DACs	参考电流

9,10 －(B－Y),－(R－Y)	色差信号转换的色度模拟输出
11 (Y0)	亮度模拟输出
13 (LL3/LL1.5)	时钟输入，信号扫描并行模式，f＝13.5MHz 或双扫描并行模式 f＝27MHz
14 (Vdd)	电源输入
15 (LL2)	(数据字 D0－6 时钟输入)只为串行数据，f＝20.25MHz，低版本。当选择并行数据格式时不要接。
16－19 (UV0－3)	数字色度输入，f＝13.5MHz 或者 7MHz
20 (Vssd)	数字地
21－27 (D0－6)	数字 7 位亮度输入，f＝13.5MHz 或 27MHz
28 (BL/)	格式输入，标志一个数据线的传送开始

3.7.1 功能描述

VDA，DMSD/S，DMSD，和 RGB 控制器构成一个数字 TV 系统的视频统道。VDA 从 DMSD/S，DMSD 接收亮度和色度数据并把这些数据转换成 RGB 控制器输出的模拟信号。

色度数据信号

色度数据包含由采集频率为 3.375MHz(单扫)的交替 UV 采样值，使用两个级联插入过滤器把采样频率增加到 13.5MHz。然后 7 位色度数据转换成用于 RGB 控制器输出的模拟信号(转换色差信号 B－Y 和 R－Y)。

亮度数据信号

亮度数据频率被锁定在 13.5MHz 或 27MHz，进入分辨率增强过滤器(由 CTRL0 和 CTRL1 控制)，这改善了小变量区的量化噪声行为并产生 8 位数据输出。这 8 位数据转换成用于 RGB 控制器输出的模拟信号。

BL/信号

BL/信号用于标志行内激活视频长度，并同步 UV 数据的多路。

操作模式

有两种操作模式：

并行数据传送(单扫)；LL3/LL1.5＝13.5MHz

并行数据传送(双扫)；LL3/LL1.5＝27MHz

操作信号

输出信号被 AC 偶合到 RGB 控制器。在水平同步的间隙亮度和色度信号消隐(黑或无色差)，并且 RGB 控制器钳位输入信号。

框图见图 3.7：

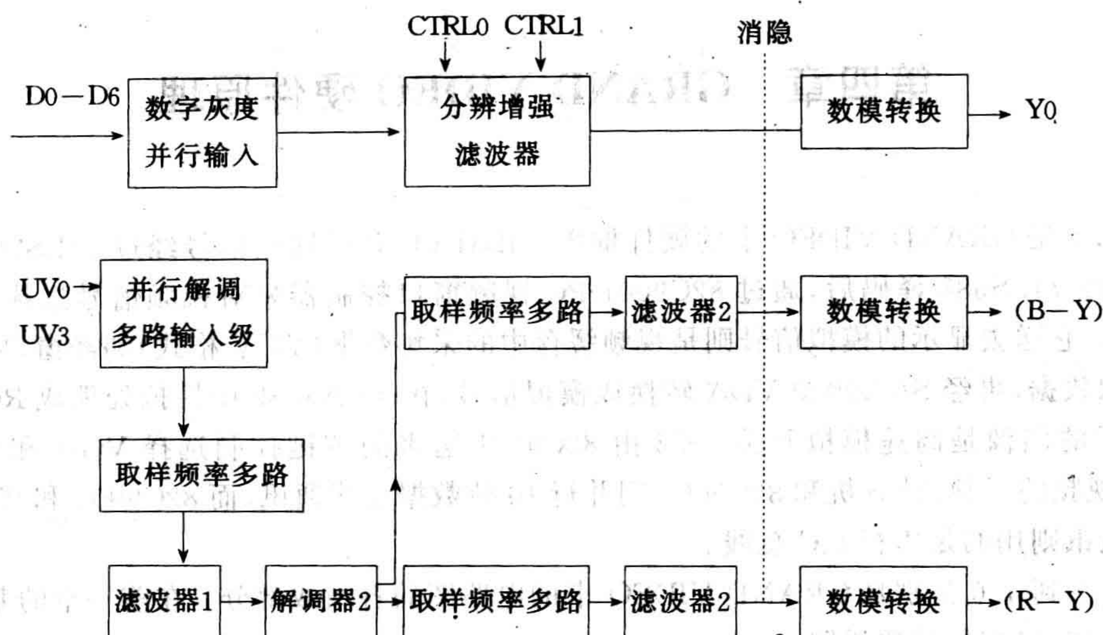


图3.7 SAA9060方框图

第四章　GRAND VIDEO 硬件原理

图 4.1 是 GRAND VIDEO 卡的硬件框图。IBMAT 总线数据信号经过 74LS245 双相缓冲，地址经 74LS688 译码后，通过 82C9001 PC 视频窗口控制器来对视频信号选择，处理，俘获，存取。它送去显示的模拟信号则是视频缓存中的采集数据(YUV 格式)和经窗口和色键处理的图像数据，再经 SAA9060 VDA 转换成模拟信号，在 TDA4680 中模拟处理成 RGB 信号。最终显示的图像是高速模拟开关 4053 由 82C9001 送来的色键控制选择 VGA 和 GRAND-VIDEO 视频的一种。计算机和 82C9001 用并行 16 位数据总线通讯，而 82C9001 和其它 Philips 芯片的通讯则用的是串行 I2C 总线。

图 4.2 到 4.6 分别是 GRAND VIDEO 卡的电路图的各组成部分。在前一章的基础上，对个分电路图，只需作简单说明。

图 4.2 是 AT 总线驱动和译码部分。16 位双向数据总线用两片 74LD245 驱动。译码采用一片 74LS682 八位数字比较器。其余信号直接供给 SAA9051 等芯片。图 4.3 是数字视频的前级。这里 TDA8708 进行复合视频信号的视频模拟数字转换，视频源的选择(I0 和 I1 管脚)和采样时钟(管脚 5)由 SAA9051 多制式译码器提供，后者对复合视频输入信号(CVBS0－7)进行多制式译码成灰度(Y0－7)和色差(UV4－7)信号及同步信号(HS 和 VS)供 82C9001 使用。它的时钟 LL3 输出经 SAA9057 和 74H04 延时作为像素时钟 PIXCLK。编程 I2C 总线信号由 SDA，SCL 完成。图 4.4 是 82C9001 视频窗口控制器部分。它是 GRANDVIDEO 的核心芯片。它的左端的 Y0－7 和 UV4－7 是已译码的数字视频信号输入，下方的 A0－23，D0－15 等来自 CPU AT 总线，右端的 P0－7 来自 VGA 特征数字插座。上左方是经窗口变换后待输出到视频缓存器存储的数据，多路的地址线则由 MA0－7 和 RASY，RASC，CAS0－1 等担任。最后，SAA9001 还需挑起 I2C 总线主设备的"大梁"，I2C 总线的硬件相当简单，这里是由 R17 和 R15 联结正 5 伏完成，I2C 总线所属设备只需把 SCL 和 SDA 各自联结即可。但是它的编程比并行总线要复杂得多(参看第七和九章)。82C9001 还送到 4053 模拟开关一色键控制信号 VIOKEY。图 4.5 是视频缓存器部分。一共用了 6 片 4 位 256K 的并行写入(W/IO0－3)和串行移位输出(SIO0－3)的 RAM 芯片。其中左方 4 片存储灰度数据，右方两片存储色度信号。74ALSLS574 和 74ALS257 把输出像素数据组织成适应于 SAA9060 需要的方式。图 4.6 是数字视频的末级。SAA9060 把三路数字视频信号(YUV)转换成三路模拟视频信号，TDA4680 把它们转变成 RGB 信号，并对峰白及黑电平自动控制，经 I2C 总线再处理。GRANDVIDEO 俘获视频图像和 VGA 图形的色键功能由 4053 经连在一起的 A，B，C 脚实现，它们由 82C9001 的 VIOKEY 送来。最后，混合模拟信号经三路功放(AD847)送显示器。

图 4.7 是它的元件在卡上的分布图。

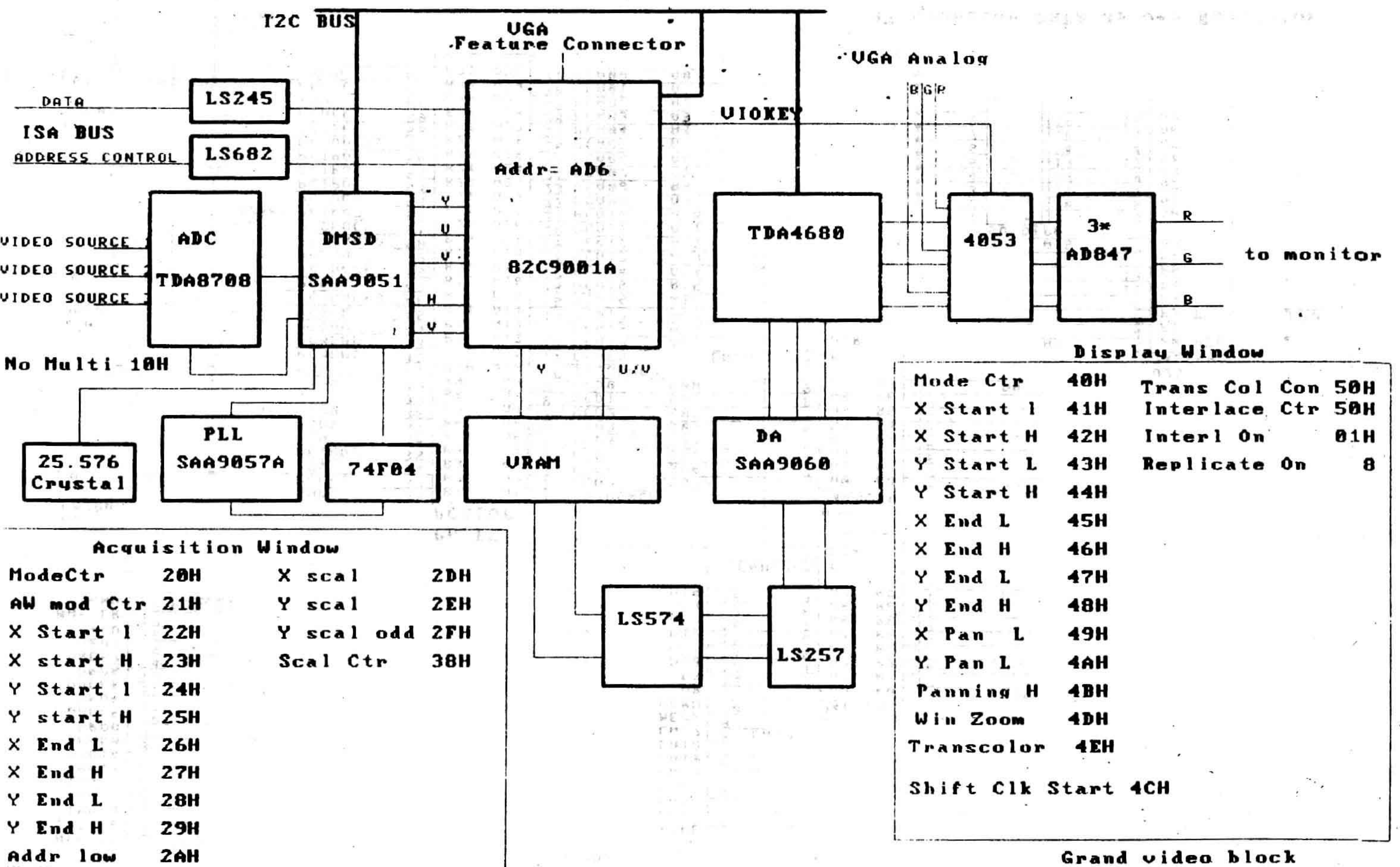


图 4.1　GRAND VIDEO YUV 卡的框图



图 4.2 GRAND VIDEO 卡电路图:AT 总线驱动和译码

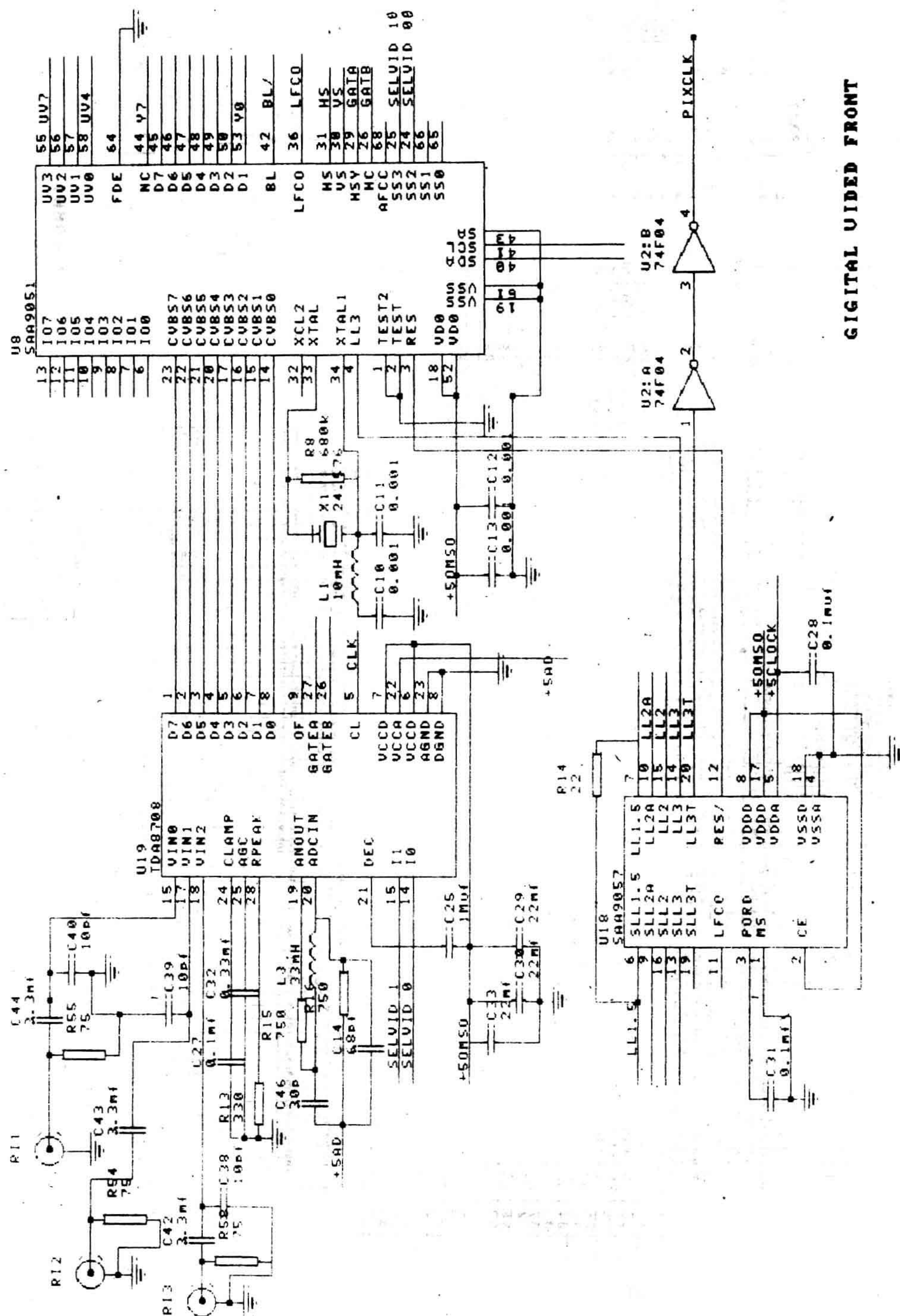


图 4.3　GRAND VIDEO 电路图：数字视频前级

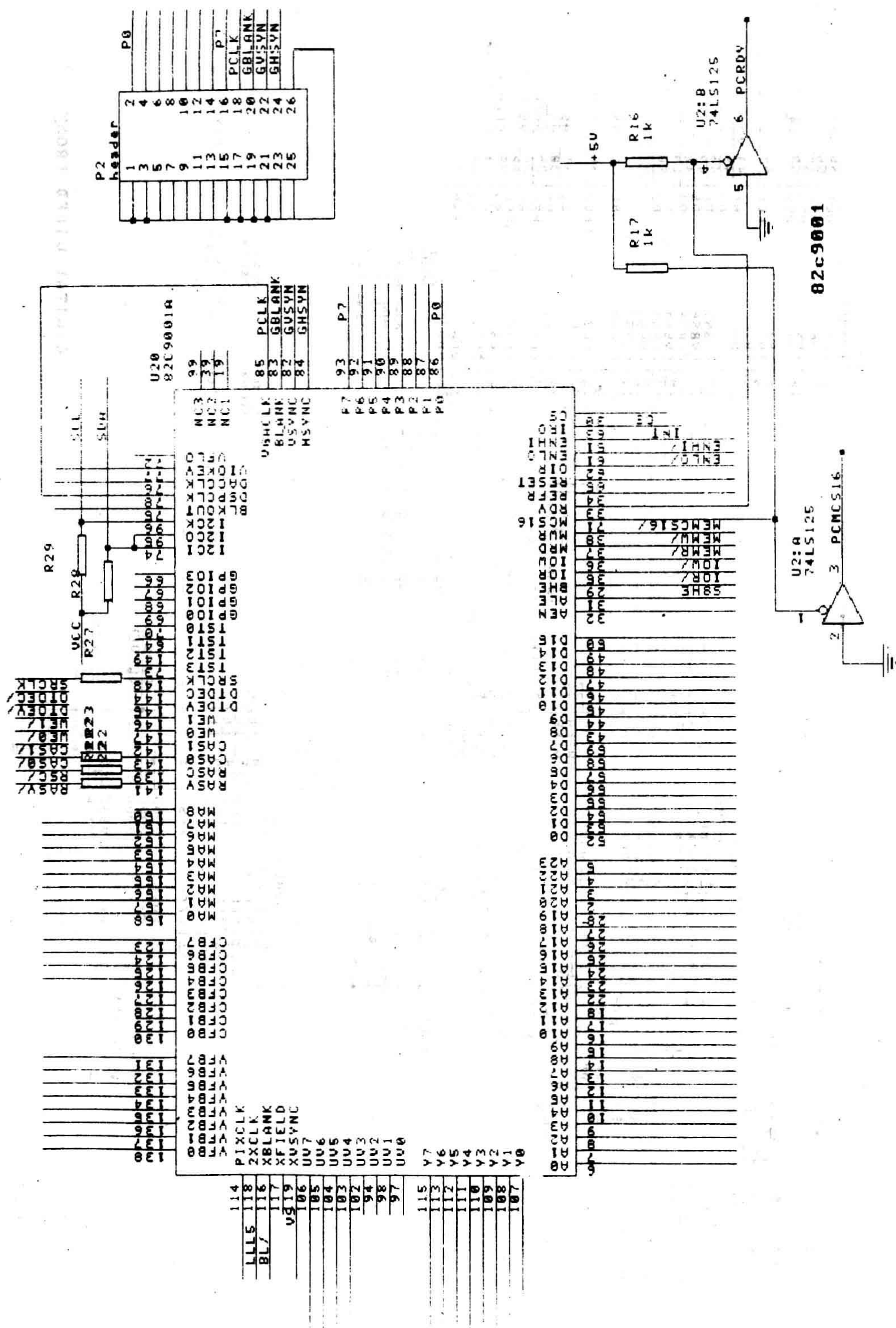


图 4.4　GRAND VIDEO 电路图：SAA9001 视频窗口控制器

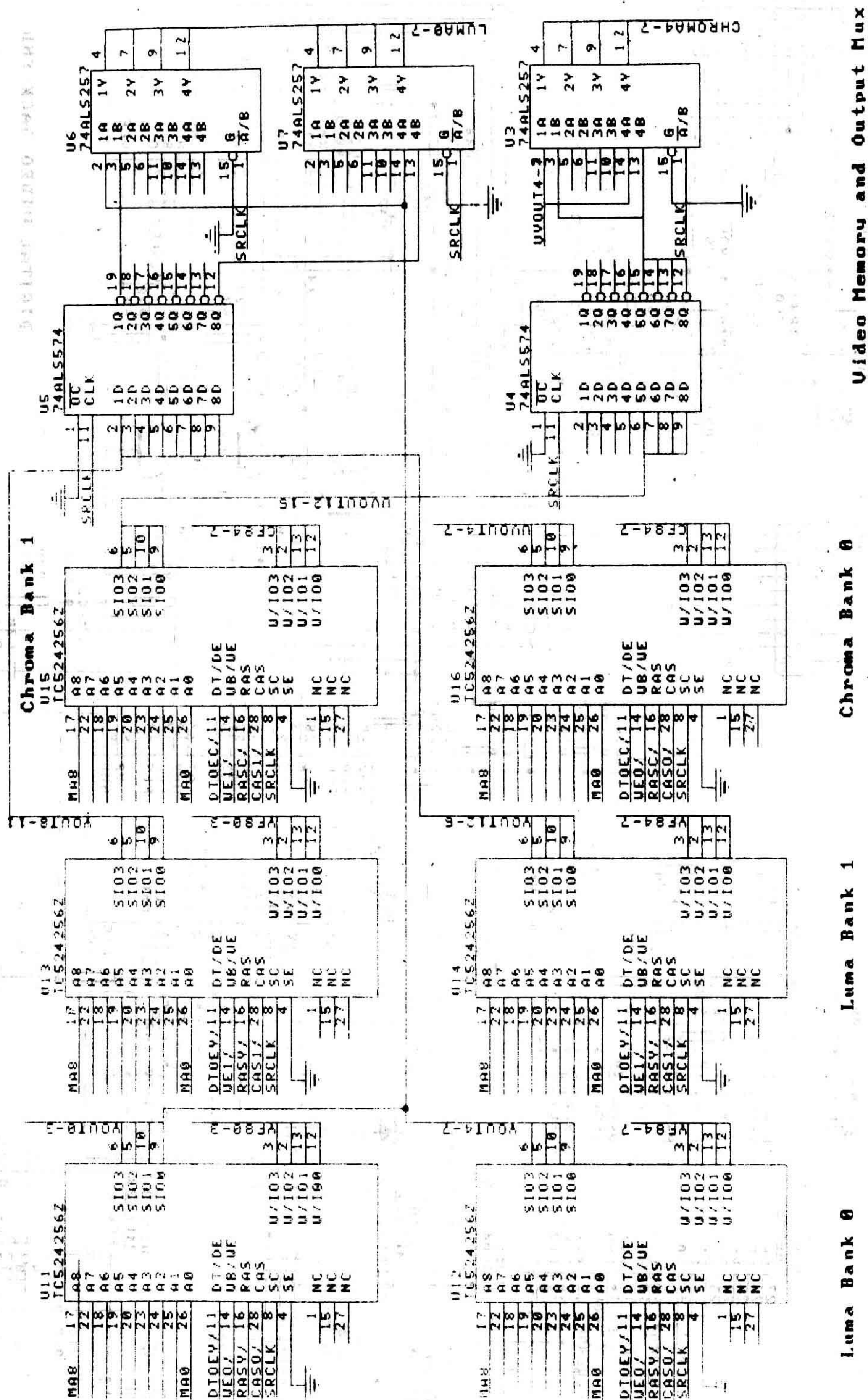


图 4.5 GRAND VIDEO 电路图:视频缓存器

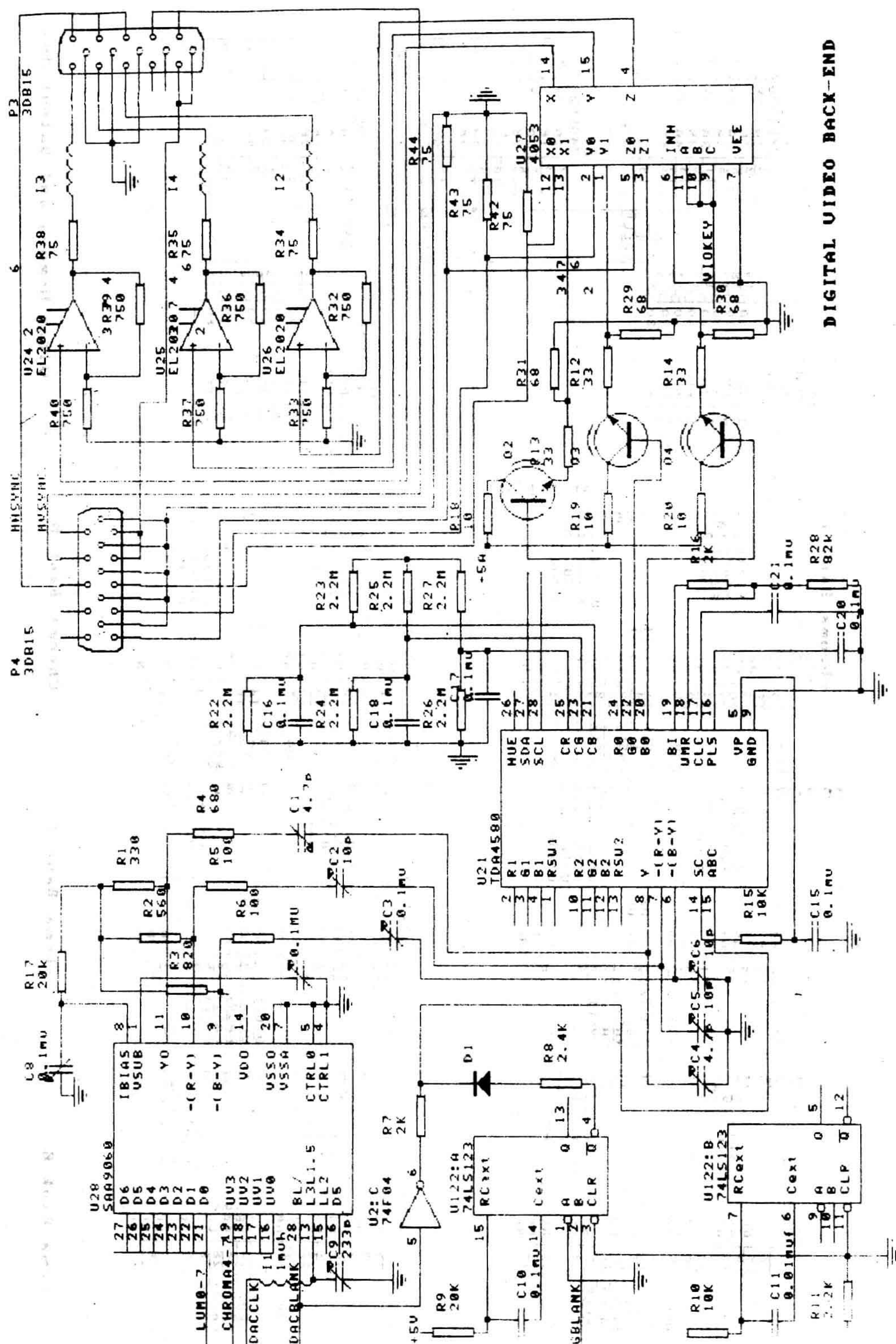
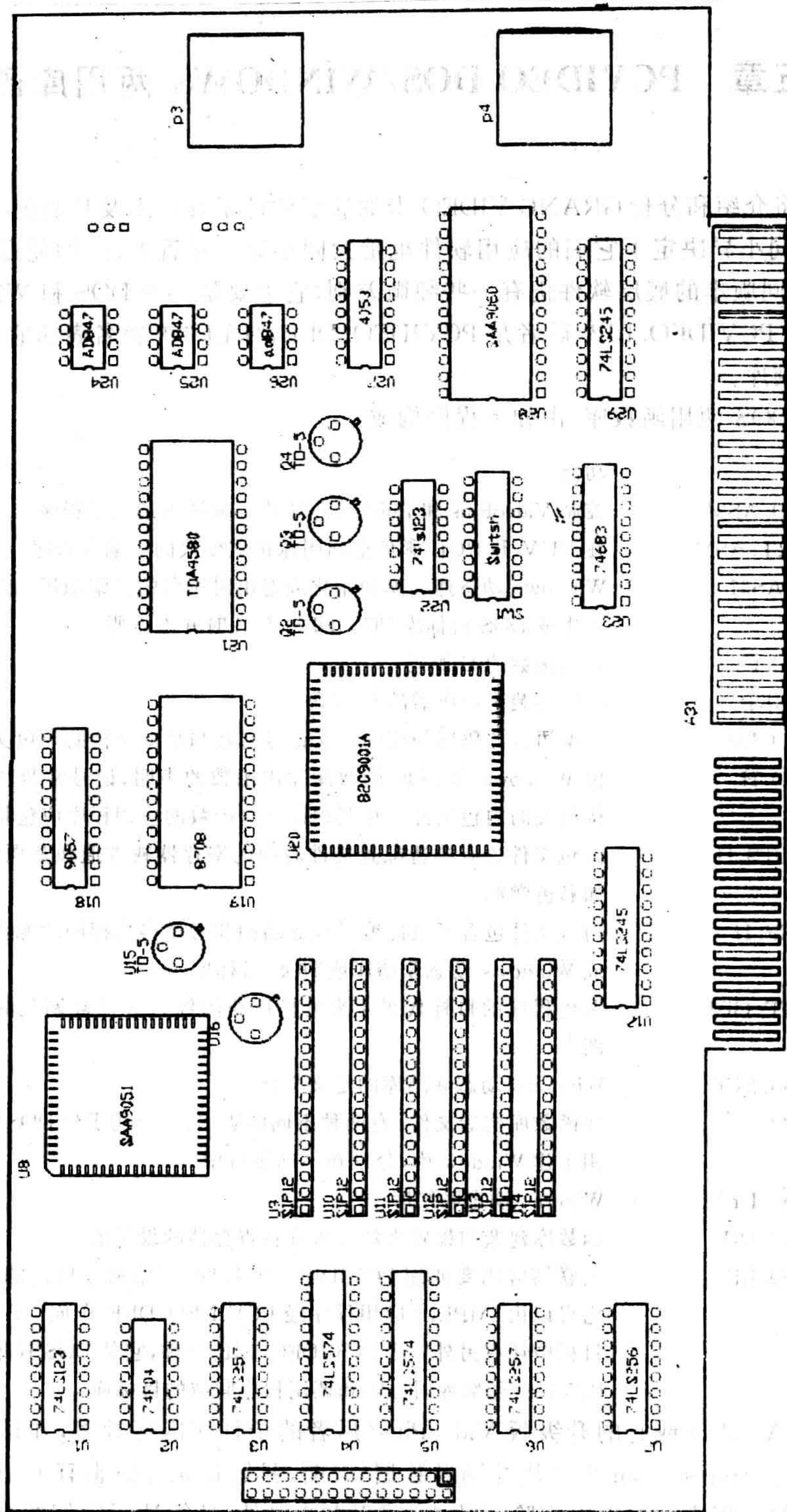


图 4.6　GRAND VIDEO 电路图：数字视频末级



GRANDPCB Check Plot

图 4.7 GRAND VIDEO 卡元件分布图

第五章　PCVIDEO DOS/WINDOWS 两用库函数

以下五章将介绍和分析GRAND VIDEO卡的底层软件子程序库及其源码。各种多媒体视频卡的硬件大同小异决定了它们的应用软件也是大同小异。尽管如此，即便是同一GRAND VIDEO卡的不同版本的底层软件仍有一些细微差别，它主要是一个DOS和WINDOWS两用函数库，前者是PCVIDEO.LIB，后者是PCVIDEO.DLL，"DLL"扩展名表明它是WINDOWS专用的动态连结库。

Windows/DOS两用函数库，由如下程序构成：

程序名	功能
PCVIDEO.ASM	控制VideoPlus和Phillips芯片的汇编源码杂项子程序
CONVERT.ASM	在YUV和RGB彩色空间图像格式变换的汇编源程序
LIBINT.ASM	Windows动态连结库初始化或退出时所用的汇编源码。此文件对于生成DOS目标库PCVIDEO.LIB时并不需要
PCVIDEO.C	杂项函数的C源码
FRAMEGRB.C	图像存装函数所需的C源码
PCVIDEO.INC	汇编语言源码所使用的定义记号常数和数据结构的内包文件
PCVIDEO.H	由Windows和DOS函数库输出函数的声明，记号常数和数据结构相关的内包文件。凡调用任一这函数的应用程序应包括这文件
DEFAULTS.H	内包文件含有构造配置文件时用的寄存器缺省值和杂项设定(例如彩色颗粒)
TYPEDEF.H	内包文件包含了条件编译数据结构和宏使得从相同的源程序来组成Windows动态联结库或DOS目标库
PALETTE.OBJ	调色程序的目标代码用来变换俘获图像为8位被调色的彩色点图
PCVIDEO.DEF	Windows动态联结库的定义文件
PCVIDEO.	建函数库构造文件。有两种不同的版本：一个用于建DOS库；另一用于建Window库，分别在不同子目中
PCVIDEO.DLL	Window动态联结库
PCVIDEO.INI	函数库建造的配置文件用来存各寄存器的设定值
PCVIDEO.LIB	当联结应用要调用PCVIDEO.DLL时，联结命令所需的库文件。它可以由IMPLIB应用程序经PCVIDEO.DEF作成。在DOS子目栏中还有另外一个PCVIDEO.LIB文件，它是DOS程序在调用PCVideo函数库时必需和调用程序联结的目标库

在分析PCVIDEO或它的升级版VideoPlus(后者的子程序内容较多，并且还包含了新的子程序。例如Compress.asm用于压缩图像数据)以前，我们首先应熟悉有关功能库函数的各直接面向应用的子程序。以下内容除了功能，调用方法以外，还简述了它们的纵向和横向的内外联系。按子程序英文名字首的字母次序排列，凡是程序名前有 * 的表示该子程序比较简单，基本上只有寄存器操作。程序名上方加水平线的表示此子程序常用，后面将进一步介绍。此

外，我们还按子程序的有无输入输出参数加以分类。

以下程序有输入参数并且返回值：

程序名	PCV_ClearVideoRect	汇编语言
造句法	int PCV_ClearVideoRect(xLeft,yTop,width,height)	
功能	把俘获缓存中所标明的矩形区清零	
输入参数	类别/描述	
	xLeft　int 矩形区左边像素座标	
	yTop　int 顶端像素座标	
	width　int 矩形宽度(像素点数)	
	height　int 高度	
返回值	1　成功	
	0　写视频缓存出错	
备注	应确保 PC Video 缓存不和系统内存矛盾	
建议参考	PCV_SetVideoAddress	
所调用的子程序：		
	FreezeVideo	
	UnFreezeVideo	
	ComputeVideoAddress	
	Compute LoopParameters	
	BufferToExtended	

---

* 程序名	PCV_GetColor	C 语言
造句法	void PCV_GetColor(iColorIndex)	
功能	此函数返回设定的彩色级别	
输入参数	类型/描述	
	iColorIndex　int 彩色指针(以下指针有效)	
	彩色指针	
	BRIGHTNESS	
	SATURATION	
	CONTRAST	
	HUE	
	RED	
	GREEN	
	BLUE	
返回值	色度值(0—255)	
可参看	PCV_SetColor, PCV_ResetColors.	
调用子程序：	无	

---

* 程序名	PCV_GetSkewFactor	C 语言
造句法	word PCV_GetSkewFactor(iAlignmentFactor)	

功能	返回所标明的调准设定. 这些与时序相关的参数随安装的 VGA 卡和图形方式而变	
输入参数		类型/描述
	iAlignmentFactor	int 有效的指针值如下
	指针	描述
	SF_DISPWINSKEWX	显示窗 X skew
	SF_DISPWINSKEWY	显示窗 Y skew
	SF_DISPADDRSKEWX	显示地址 Y skew
	SF_DISPADDRSKEWY	显示地址 Y skew
	SF_SHIFTCOLOKSTART	彩色移位开始
	SF_PALETTESKEW	调色板 skew
	SF_PLLDIVISOR	锁相环分频系数
返回值	所标明的歪斜因子	
备注	多数板没有锁相回路	
可参看	PCV_SetSkewFactor, PCV_ResetSkewFactors	
调用子程序:	无	

---

* 程序名	PCV_GetSystemMetrics	
造句法	WORD PCV_GetSystemMetrics(wMetric)	C 语言
功能	返回所标明的信息	
输入参数		类型/描述
	wMetric	WORD 表明所需要的信息 有效类型表列如下
	SM_VIDEOWIDTH	俘获视频宽度
	SM_VIDEOHEIGHT	俘获视频高度
	SM_BOATDTYPE	板执行方式检查. 有效类型是 BT_RGB 16 和 BT_YUV411
	SM_VERSION	Scilicon 版本号
	SM_INTERLACE	如果隔行扫描输出就返回数值=INTERLACE_ON. 否则为零
	SM_REPLICATE	如果帧复制允许, 返回数值=REPLICATE_ON, 否则为零
	SM_IMAGEWIDTH	用 PCV_LoadImageRect 命令装入图像的宽度
	SM_IMAGEHEIGHT	图像高度
	SM_IMAGETYPE	图像类别 有效种类如下:
	BM_YUV411	IBM MMotion (YUV)格式
	BM_DIB24	Windows 3.0 24 位设备无关位图格式 (DIB)
	BM_DIB8G	Windows 3.0 DIB 8 位灰度指标

功能	BM_DIB8P	Windows 3.0 DIB 8 位彩色(经调色的)
	BM_TRG24	Targa 24 位格式
	BM_TRG16	Targa 16 位格式

返回值　　请求的信息字

调用子程序:　　无

---

*程序名　　PCV—GetRegister　　C 语言

造句法　　BYTE PCV_GetRegister(windex)

功能　　返回用 windex 指针表示的 PC Video 或 Phillips 寄存器的值

输入参数　　类型/描述

windex　　WORD 如果为 0，则读 VideoPlus 否

则它代表 Phillips 芯片的 I2C 地址.低字节是要读的寄存器的指针

返回值　　寄存器的值

备注　　Phillips 芯片上的参数芯片是不可读的。欲知详情参看 Phillips/Signetics Video Handbook

可参看　　PCV_SetRegister

调用子程序:　　无

---

*程序名　　PCV_GetIRQUsed

造句法　　int PCV_GetIRQUsed(int IRQ)　　C 语言

功能　　设定所申请的中断号并返回中断号

参数　　待设定中断号

返回值　　中断号

调用子程序:　　无

程序名　　PCV_LoadClipboardFormat　　C 语言

造句法　　int PCV_LoadClipboardFormat(hWnd,XPos,YPos)

功能　　把剪裁板的图像按需要格式转换后存入视频缓存(Windos 应用)

输入参数

	类型/描述
hWnd	
xPos	int 块左边像素座标
yPos	int 块顶端像素座标

返回值

1	成功
0	写视频缓存出错
−1	没有剪裁板
−2	不支持的剪裁板格式
−3	剪裁板数据出错
−4	不能分配内存

	−5	内存不能固定
	−6	不认识的的文件格式

程序名	PCV_LoadImageRect	C 语言 P. 28
造句法	int PCV_LoadImageRect(pFile,xLeft,yTop)	
功能	读所述图像文件，检查它的格式，变图像数据为视频缓存格式并写到视频缓存的 xLeft，yTop 位置	
输入参数		类型/描述
	pFile	char * 文件名字符串的指针
	xLeft	int 块左边像素座标
	yTop	int 块顶端像素座标
返回值	1	成功
	0	写视频缓存出错
	−1	打开文件出错
	−2	不能分配内存
	−3	数据不足
	−4	不认识的的文件格式
备注	应保证 VideoPlus 的缓存不和系统内存地址矛盾	
可参看	PCV_SaveImageRect,PCV_SetVideoAddress	
调用子程序：	PCV_OemInit	
	PCV_WriteImageRect	

程序名	PCV_ReadImageRect	汇编语言 P. 31
造句法	int PCV_ReadImageRect (fpBuffer, X, Y, XExt, YExt, wBitmapType, fpWorkSpace)	
功能	读图像缓存矩形区数据(5:6:5 RGB 或 4:1:1 YUV)并变成指定格式写到输出远指针	
输入参数		类型/描述
	fpBuffer	D 输出缓存的远指针
	X	W 图像块的左边座标
	Y	W 图像块的顶端座标
	XExt	W 块宽度
	YExt	W 块高度
	wBitmapType	W 位图类型
	fpWorkSpace	D 调色板工作区远指针
返回值	AX=1 成功	
调用子程序：	ExtendedToBuffer	
	ComputeVideoAddress	
	ComputeLoopParameters	

程序名	PCV_SaveClipboardFormat	C 语言
造句法	int PCV_SaveClipboardFormat（hWnd，XPos，YPos，Width，Height，wFileType，wOption）	
功能	把设备无关缓存的图像按需要格式转换后存入剪裁板(Windows 应用)	
输入参数		类型/描述
	hWnd	
	xpos	int 块左边像素座标
	ypos	int 块顶端像素座标
	width	int 块宽度
	height	int 块高度
	wFileType	int 文件名字符串的指针
	wOption	int
返回值	1	成功
	0	读视频缓存出错
	−1	建文件出错，可能是磁盘问题
	−2	不能分配内存
	−3	写文件出错，可能磁盘满
	−4	建调色板出错

程序名	PCV_SaveImageRect	C 语言 P.25
造句法	int PCV_SaveImageRect（pFile，xLift，yTop，width，heigh，wFileType，wOptions）	
功能	从视频缓存中读一矩形像素块按标明的图像文件格式变换并写盘	
输入参数		类型/描述
	pFile	char * 指向文件名字符串的指针
	xLeft	int 块左边的像素座标
	yTop	int 块顶端的像素座标
	width	int 以像素点数为单位的块宽度
	height	int 以像素点数为单位的块高度
	wFileType	WORD 文件类型
	文件类型	描述
	BM_YUV411	IBM MMotion (4:1:1 YUV) 格式
	BM_DIB24	Windows 3.0 24 位设备无关位图格式(DIB)
	BM_DIB8G	Windows 3.0 DIB 8 位灰度指标
	BM_DIB8P	Windows 3.0 DIB 8 位彩色(经调色的)
	BM_TRG24	Targa 24 位格式
	BM_TRG16	Targa 16 位格式
	BM_TRG16	Targa 16 bit format
	wOptions	WORD 杂项选用参数（必需为零）
返回值	>0	原来图像中的色彩数
	0	读视频缓存出错

	−1	建文件出错
	−2	分配文件出错
	−3	写文件出错
备注	应保证 VideoPlus 的缓存不和系统内存地址矛盾	
调用子程序:	FreezeVideo UnFreezeVideo AllocateMemory FreeMemory PCV_ReadImageRect	

程序名	PCV_WriteImageRect	汇编语言 P. 33
造句法	int PCV_WriteImageRect (fpBuffer, fpPalette, X, Y, XExt, YExt, wBitmapType)	
功能	从输入缓存读指定格式图像数据转换成 16 位 RGB 或 4:1:1 YUV 到视频缓存	
输入参数		类型/描述
	fpBuffer	D 输入缓存的远指针
	fpPalette	D 调色板工作区远指针
	X	W 图像块的左边座标
	Y	W 图像块的顶端座标
	XExt	W 块宽度
	YExt	W 块高度
	wBitmapType	W 位图类型
返回值	无	
调用子程序:	ComputeVideoAddress ComputeLoopParameters BufferToExtended	

以下程序有输入参数,但不返回值

程序名	PCV_CreatWindow	汇编语言 P. 13
造句法	void PCV_CreateWindow (xLeft, Top, xExtent, yExtent, fFitToWindow)	
功能	在 VGA 给定位置建一个视频显示窗	
输入参数		类型/描述
	xLeft	int VGA 显示器上视频窗左边位置
	yTop	int 顶端位置
	xExtent	int 宽度
	yExtent	int 高度
	fFitToWindow	WORD 旗用来表示视频窗是否要和显示窗匹配 '1' = 匹配显示窗

'0' ＝ 要视频全部并对它裁剪

返回值	无	
可参看	PCV_SetWindowSize，PCV_SetWindowPosition， PCV_PanWindow	
调用子程序：	updateVideoWindow FreezeVideo UnFreezeVideo， WaitVGAVerticalRetraceStar HorizentalZoom VerticalZoom unScaleVideo WaitToAcquireField	

程序名	PCV_HorizontalZoom	汇编语言　P. 14
造句法	void PCV_HorizontalZoom(iZoomFactor)	
功能	设定水平放大因子. 俘获视频可以在输出时放大 2,4,或 or 8 倍.	
输入参数	类型/描述	
	iZoomFactor　int 放大因子可以从 0 到 3. 0=1 倍，1=2 倍，2=4 倍，and 3=8 倍.	
返回值	无	
备注	放大在 VideoPlus 版本 1 中不支持.	
可参看	PCV_VerticalZoom	
调用子程序：	HorizontalZoom UpdateVideoWindow	

---

* 程序名	PCV_SetColorKey	汇编语言　P. 41
造句法	void PCV_SetColorKey(wColorIndex)	
功能	设定彩色指数作为视频显示的彩色键. VGA 显示缓存中任何一个设为这个颜色的像素将显示从视频缓存中来的信号.	
输入参数	类型/描述	
	wColorIndex　int 表示彩色指数用作选择视频缓存和 VGA 显示.	
返回值	无	
调用子程序：	无	

程序名	PCV_SetColor	C 语言
造句法	void PCV_SetColor(iColorIndex,iColorValue)	
功能	设定彩色级别.	
输入参数	类型/描述	

	iColorIndex	int 彩色指针(有效针如下)
	彩色指针	
	BRIGHTNESS	
	SATURATION	
	CONTRAST	
	HUE	
	RED	
	GREEN	
	BLUE	
	iColorValue	int 色码在 0 和 255 之间.
返回值	无	
可参看	PCV_GetColor,PCV_ResetColos	
调用子程序:	wr_ir2c	

程序名	PCV_PanWindow	汇编语言 P.18
造句法	void PCV_PanWindow(xLeft,yTop)	
功能	设定视频窗中的位置.	
输入参数		类型/描述
	xLeft	int 视频缓存器中待显示区的左边位置(像素单位)
	yTop	int 视频缓存器中待显示区的顶端位置(像素单位)
返回值	无	
备注	函数只当视频是满尺度时方起作用. 视频是否按窗大小或满尺度显示是由 PCV_CreateWindow 和 PCV_SetWindowSize 控制. 它对在 VGA 中显示窗的位置和大小没有影响. 它只改变视频缓存在窗中显示的那一部分.	
可参看	PCV_CreateWindow ,PCV_SetWindowSize, PCV_SetWindowPosition	
调用子程序:	SetCropVideoRect WaitVGAVerticalRetraceStart SetDisplayPosition	

---

* 程序名	PCV_SetAcquisitionWindow P.43	
造句法	void PCV_SetAcquisitionWindow(X1,Y1,X2,Y2)	
功能	设定存取窗口	
输入参数		类型/描述
	X1	int X 开始座标
	Y1	int Y 开始座标
	X2	int X 结束座标
	Y2	int Y 结束座标

返回值	无	
调用子程序：	SetAcquisitionWindow	P. 43

---

* 程序名	PCV_SetCaptureAddress		P. 42
造句法	void PCV_SetCaptureAddress(XX,YY)		
功能	设定俘获视频在缓存中的地址		
输入参数		类型/描述	
	XX	int X 座标	
	YY	int Y 座标	
返回值	无		
调用子程序：	SetCaptureAddress		

程序名	PCV_SetDisplayWindow		P. 16
造句法	void PCV_SetDisplayWindow(X,Y,XExt,Yext)		
功能	设定显示窗寄存器		
输入参数		类型/描述	
	X	int X 开始座标	
	Y	int Y 开始座标	
	XExt	int 窗宽度	
	YExt	int 窗高度	
返回值	无		
调用子程序：	WaitVGAVerticalRetraceStart		
	SetDisplayWindow		

程序名	PCV_SetInputFormat		P. 45
造句法	void PCV_SetInputFornat(iVideoFormat)		
功能	设定输入视频制式.		
输入参数		类型/描述	
	iVideoFormat	int 视频制式(CF_NTSC 或 CF_PAL)	
返回值	无		
可参看	PCV_GetInputFormat		
调用子程序：	wr_I2c		
	SetAccquisitionWindow		

---

* 程序名	PCV_SetPortAddress		-C 语言
造句法	void PCV_SetPortAddress(iPortAddr)		
功能	设定 VideoPlus 口地址.		
	输入参数	类型/描述	
	iPortAddr	int VideoPlus 口地址	

返回值　　无

可参看　　PCV_GetPortAddress

调用子程序:

---

*程序名　　PCV_SetRegister　　C 语言

造句法　　void PCV_SetRegister(WORD windex,BYTE bValue)

功能　　设定所述 VideoPlus 或 Phillips 寄存器值.

输入参数

	类型/描述
windex	WORD 如果高字节等于 0,写到 VideoPlus. 否则,这字节表示要写的 Philips 芯片的 I2C 地址. 低字节是要写的寄存器的指针.
bValue	BYTE 待写的值

返回值　　无

可参看　　PCV_GetRegister

调用子程序:　　无

程序名　　PCV_SetSkewFactor　　C 语言

造句法　　void PCV_SetSkewFactor(iAlignmentFactor, wValue)

功能　　设置所需的视频调节因子. 这参数和所安装的 VGA 卡的图形方式时序有关.

输入参数

	类型/描述
iAlignmentFactor	int 有效的指针值如下
指数	描述
SF_DISPWINSKEWX	显示窗 X skew
SF_DISPWINSKEWY	显示窗 y skew
SF_DISPADDRSKEWX	显示地址 X skew
SF_DISPADDRSKEWY	显示地址 Y skew
SF_SHIFTCLOCKSTART	显示时钟开始
SF_PALETTESKEW	调色板 skew
SF_PLLDIVISOR	锁相环分频系数
wValue	int 调节系数的新值.

返回值　　无

备注　　多数板没有锁相环.

可参看　　PCV_GetSkewFactor, PCV_ResetSkewFactors

调用子程序:　　UpdateVideoWindow

程序名　　PCV_SetVideoAddress

造句法　　void PCV_SetVideoAddress(iVideoAddr)

功能　　设定视频缓存的物理地址(0—15)

输入参数		类型/描述
	iVideoAddr	int VideoPlus 图像缓存以兆字节表示的地址(1—15).
返回值	无	
可参看	PCV_GetVideoAddress	
调用子程序:	对 DOS 无 对于 WINDOWS 有 FreeSelectors AllocNInitSelectors	

程序名	PCV_SetVideoMode	
造句法	void PCV_SetVideoMode(wFlag)	
功能	设定视频裁剪模式	
输入参数		类型/描述
	wFlag	int 视频裁剪模式旗
返回值	无	
调用子程序:	SetAcquisitionWindow DisableScaling SetCropVideoRect DisableScaling	

程序名	PCV_SetVideoScaling	P. 16
造句法	void PCV_SetVideoScaling(xSize,ySize,fFitInWindow)	
功能	设定视频标定并且选定标定或否	
输入参数		类型/描述
	XSize	int 窗宽度
	YSize	int 窗高度
	wFitInWindow	int 是否标定的旗 1: 标定 0: 不标定
返回值	无	
调用子程序:	WaitVGAVerticalRetraceStart FreezeVideo VerticalZoom DisableScaling EnableFieldReplicate ScaleVideo WaitToAcquireField	

---

* 程序名	PCV_SetVideoSource
造句法	Void PCV_setVideoSource(wSourceSelect)

功能	设定视频信号源	
输入参数		类型/描述
	xSourceSelect)	int 所选的视频信号源
返回值	无	
调用子程序:	SetVideoSource	
	wr_i2c	P. 46

程序名	PCV_SetWindowPosition	P. 13
造句法	void PCV_SetWindowPosition(xLeft,yTop)	
功能	设定在 VGA 上视频显示窗的左顶角位置	
输入参数		类型/描述
	xLeft	int 在 VGA 上显示的视频窗的左边用像素表示的位置
	yTop	int 在 VGA 上显示的视频窗的顶端用像素表示的位置
返回值	无	
可参看	PCV_CreateWindow, PCV_SetWindowSize, PCV_PanWindow	
调用子程序:	UpdateVideoWindow	

程序名	PCV_SetWindowSize	P. 13
造句法	void PCV_SetWindowSize(xExtent,yExtent,fFitToWindow)	
功能	设定在 VGA 上视频显示窗的宽度和高度	
输入参数		类型/描述
	xExtent	int 用像素数表示的窗宽
	yExtent	int 用像素数表示的窗高
	fFitToWindow	word 表示要否把视频和窗匹配的旗. '1' =和窗相匹配 '0' =保持视频满屏并裁剪多余部分
返回值	无	
可参看	PCV_CreateWindow, PCV_SetWindowSize	
调用子程序:	UpdateVideoWindow	

---

* 程序名	PCV_SetWriteProtecMask	C 语言 P. 42
造句法	void PCV_SetWriteProtectMask(wMask)	
功能	设定采集视频时用来写保护视频帧缓存中某位的掩码	
输入参数		类型/描述
	wMask	WORD 16 位掩码. 等于 0 的位表示该位写保护
返回值	无	

备注	在板的 YUV 执行方式时，高字对应于色度位，低字对应于灰度位。对于 RGB 板，5 个最低位是蓝，它们后是 5 位绿，然后是 5 位红。最高位是绿的最低位（绿是 6 位）
可参看	VideoPlus 帧缓存格式有关章节
调用子程序：	无

程序名	PCV_VerticalZoom	汇编语言　P.14
造句法	void PCV_VerticalZoom(iZoomFactor)	
功能	设定垂直放大因子 俘获视频可在输出时放大 1，2，4，或 8	
输入参数	类型/描述	
	iZoomFactor　　int 放大因子从 0 到 3 0=1x，1=2x，2=4x，和 3=8x	
返回值	无	
备注	VideoPlus 版本 1 不支持放大	
可参看	PCV_HorizontalZoom	
调用子程序：	VertcalZoom UpdateVideoWindow	

以下程序无输入参数，返回一值

* 程序名	PCV_GetPortAddress	C 语言
造句法	int PCV_GetPortAddress()	
功能	返回 VideoPlus 口地址	
返回值	口地址	
可参看	PCV_SetPortAddress	
调用子程序：	无	

---

* 程序名	PCV_GetVideoAddress	C 语言
造句法	int PCV_GetVideoAddress()	
功能	返回视频缓存地址 无参数	
返回值	视频缓存地址(1 to 15)	
可参看	PCV_SetVideoAddress	
调用子程序：	无	

---

* 程序名	PCV_GetVideoSource	汇编语言　P.46
造句法	int PCV_GetVideoSource()	
功能	求视频信号源	
返回值	AX=设定的源	

调用子程序：	无	

---

* 程序名	PCV_GetInputFormat	汇编语言 P. 45
造句法	int PCV_GetInputFormat()	
功能	返回输入视频格式 无参数	
返回值	CF_NTSC 或 CF_PAL	
可参看	PCV_SetInputFormat	
调用子程序：	无	

程序名	PCV_Initialize	汇编语言 P. 1
造句法	int PCV_Initialize()	
功能	从 PCVIDO. CFG 装入配置信息，检测所用的图形模式 检测 VideoPlus 板的执行方式，并且进行相应的 VideoPlus 和 Phixel 寄存器初始化 无参数	
返回值	1=成功　　0=失败	
可参看	PCV_Exit，PCV_LoadConfiguration	
调用子程序：	LlLoadLibrary GetProcAddress PCV_LoadConfigration GetVideoMode SyncDetect SetVideoSource SetInputFormat AllocNInitSelect PCV_OemInit wr_I2c	

程序名	PCV_LoadConfiguration	C 语言 P. 6
造句法	void PCV_LoadConfiguration()	
功能	从 PCVIDEO. CFG 装入所用的配置信息。这文件包括诸如口和视频缓存地址，寄存器的设定值，视频调节和彩色级别等信息 无参数	
返回值	1　　成功 0　　文件没找到 −1　　文件数据不足. −2　　文件不是 VideoPlus 配置文件.	
可参看	PCV_Save Configuration，PCV_Intialize	
调用子程序：	GetProfileString	

GetProfileInt

程序名	PCV_SaveConfiguration	C 语言　P. 8
造句法	int PCV_SaveConfiguration()	
功能	把目前的配置存到 PCVIDEO. CFG 文件中。它包括诸如口和视频缓存地址,寄存器设定值视频调节彩色电平等信息 无参数.	
返回值	1　　成功 0　　建文件出错,磁盘可能满了 -1　　写文件出错,磁盘满	
可参看	PCV_LoadConfiguration, PCV_Exit	
调用子程序:	DOS 写文件通用程序	

下子程序较为简单,无输入及输出参数:

* 程序名	PCV_DisableColorKey	汇编语言　P. 41
造句法	void PCV_DisableColorKey()	
功能	彩色键不能	
调用子程序:	无	

程序名	PCV_DisableFieldReplication	汇编语言　P. 15
造句法	void PCV_DisableFieldReplication()	
功能	不准帧复制	
备注	VideoPlus 版本 1 不支持帧复制。仅当输入视频不标定时才允许帧复制。这种方式下仅俘获一帧,每次扫描显示二次,来减少闪烁效应	
可参看	PCV_EnableFieldReplicate, PCV_GetSystemMetrics	
调用子程序:	DisableFieldReplication VerticalZoom UpdateVideoWindow	

---

* 程序名	PCV_DisableInterlace	C 语言
造句法	void PCV_DisableInterlace()	
功能	不准隔行扫描输出.	
备注	VideoPlus 版本 1 不支持割行扫描输出	
可参看	PCV_Enableinterlace, PCV_GetSystemMetrics	
调用子程序:	无	

---

* 程序名	PCV_DisableVideo	汇编语言　P. 41

造句法	void PCV_DisableVideo()	
功能	不准显示视频.	
可参看	PCV_EnableVideo	
调用子程序:	无	

---

* 程序名	PCV_EnableColorKey	汇编语言 P.41
功能	允许彩色键功能	
调用子程序:	无	

---

* 程序名	PCV_EnableFieldReplication	汇编语言 P.14
造句法	void PCV_EnableFieldReplication()	
功能	当视频微经标定时允许帧复制	
备注	VideoPlus 版本 1 不支持帧复制。仅当输入视频不标定时才允许帧复制。这种方式下仅俘获一帧,每次扫描显示二次,来减少闪烁效应	
可参看	PCV_DisableFieldReplication,PCV_GetSystemMetrics	
调用子程序:	EnableFieldReplicate	P.42

---

* 程序名	PCV_EnableInterlace	C 语言
造句法	void PCV_EnableInterlace()	
功能	允许隔行扫描输出.	
备注	VideoPlus revision 1 does not support interlacedoutput	
可参看	PCV_DisableInterlace	
调用子程序:	无	

---

* 程序名	PCV_EnableVideo	汇编语言 P.41
造句法	void PCV_EnableVideo()	
功能	允许视频帧缓存中的俘获视频显示. 对于 VGA 显示与 PCV_SetColorKey 功能相配的像素并且在由 PCV_SetWindowSize 和 PCV_SetWindowPosition(或 PCV_CreateWindow 功能)所给定的显示窗以内的每一个像素	
备注	视频显示窗和彩色键必需在这以前设置好	
可参看	PCV_DisableVideo	
调用子程序:	无	

程序名	PCV_Exit	汇编语言 P.4
造句法	void PCV_Exit()	
功能	关闭视频显示和存配置信息到 VideoPlus.CFG	

可参看	PCV_DisableVideo, PCV_Initialize, PCV_SaveCon－figuration
调用子程序：	TurnBorder
	FreeSelector

程序名	PCV_FreezeVideo	汇编语言
System	void PCV_FreezeVideo()	
功能	停止视频信号俘获。最后俘获的一帧在 VGA 显示窗下一直显示到用 PCV_UnFreezeVideo 功能解冻为止	
备注	它不影响视频缓存中视频如何显示. 它只停止俘获视频信号	
可参看	PCV_UnFreezeVideo	
调用子程序：	freezeVideo	P. 21
	WaitVideoVerticalRetraceStart	
	WaitToAcquireField	

---

* 程序名	PCV_ProfileInt	C 语言 P. 12
功能	返回从 PCVIDEO. INI 文件得来的整数键值	
返回值	无	
调用子程序：	PCV_GetProfileString	

---

* 程序名	PCV_GetProfileString	C 语言 P. 12
功能	写字符串到 PCVIDEO. INI 文件	
调用子程序：	无	

程序名	PCV_ResetColors	C 语言
造句法	void PCV_ResetColors()	
功能	恢复彩色级别到缺省值	
	无参数	
返回值	无	
可参看	PCV_GetColor, PCV_SetColor	
调用子程序：	wr_i2c	

程序名	PCV_ResetSkewFactors	C 语言
造句法	void PCV_ResetSkewFactors()	
功能	恢复所有的调节因子到缺省值	
	无参数	
返回值	无	
可参看	PCV_GetSkewFactor, PCV_SetSkewFactr	
调用子程序：	UpdateVideoWindow	

* 程序名	PCV_UnFreezeVideo	汇编语言
造句法	void PCV_UnFreezeVideo()	
功能	再次俘获视频图像。调用这功能之后活动视频图像在 VGA 窗上显示	
备注	它只能重新俘获视频。它不影响俘获视频是否或如何显示。俘获视频的显示由 PCV_EnableVideo，PCV_SetWindowPosition，PCV_SetWindow-Size 和 PCV_FreeVideo 控制.	
可参看	PCV_FreezeVideo	
调用子程序：	UnfreezeVideo	P. 44

程序名	PCV_WaitVGARetrace	汇编语言
功能	等待 VGA 垂直回扫开始	
调用子程序：	WaitVGAVerticalRetraceStart	

以上以 PCV_开头的子程序调用的公共子程序有如下一批：

AllocateMemory：	分配一块给定大小的内存块并返回一远指针
AllocNInitSelectors：	分配初始扇区
BufferToExtended：	缓存到扩充内存块传输
ComputeVideoAddress：	计算对应于 X，Y 的视频缓存地址
ComputeV-ideoLoopParameters：	计算行扫描增量和循环参数
CreateDIBPalette：	构造一个与设备无关的逻辑调色板，返回把柄
ConvertPaletteTable：	对 PCX 和 TIFF 文件格式变换调色板用表
DisAbleScaling：	停止对视频标定
EnableFieldReplication：	允许帧复制
ExtendedToBuffer：	扩充内存向缓存传送
FarMemCpy：	拷贝内存到缓存
FitVideo：	匹配视频
FreeMemory：	释放内存
FreeSelectors：	释放所有段
FreezeVideo：	冻结视频
FreezeV-ideoUnsybchronized：	非同步冻结视频
GetVideoMode：	获取 VGA 显示模式对比分辨率等参数
HorizontalZoom：	水平放大
LoadPalette：	装入调色板
RGB16ToRGB32：	RGB16 格式到 RGB32 格式的转换
……	其它图像文件格式之间转换

ScaleVideo:	标定视频
SetAcquisitionWindow:	设定视频存取窗
SetCaptureAddress:	设定俘获内存地址
SetCropVideoRect:	设定视频采集窗来裁剪视频矩形窗
SetDisplayWindow:	设定显示窗
SetDisplayPosition:	设定显示窗位置
SetInputFormat:	设定视频输入格式
SetVideoSource:	选择视频源
SyncDetect:	VGA 同步检测
TurnBorder:	纠错程序用来设定 VGA 边框颜色
UnFreezeVideo:	开始采集视频
UnScaleVideo:	停止标定视频
UpdateVideoWindow:	更新视频窗
VerticalZoom:	垂直放大
WaitToAcquireField:	等待采集完一帧
WaitVGA-HorizontalRetraceEnd:	等待 VGA 水平回扫结束
WaitVGAVer-ticalRetraceStart:	等待 VGA 垂直回扫开始
Wait-VideoVerti-calRetraceStart:	等待视频垂直回扫开始

再下一级的被调用的子程序有 DOS 中断调用，文件操作，高内存访问等，它们在 C 或 MASM 语言的内包文件 CMACRO. INC，stdio. h，stdlib. h，conio. h，io. h，string. h，direct. h，fcnt. h，sys\types. h 和 sys\stat. h 中. 这里不再列举。

第六章　PCVIDEO 软件所用内包文件和头文件

在分析 PCVIDEO 源文件以前，有必要结合前几章硬件知识来熟悉一下 MASM 和 C 语言源程序中将要用到的内包(.INC)和头(.H)文件的内容。为方便读者，我们已加上中文注释。

```
page ,132
;******************************************
;程序  :     CVIDEO.DLL
;模块  :     PCVIDEO.INC
;描述  :     PC Video 硬件接口子程序中所用的常数和结构定义常数名全用英文大写字母，结构
             变量除组成单词第一字或缩写用大写以外，全用小写表示
;版本号:     1.6
;版权  :  (C) Chips and Technologies, 公司 1991, 1992
;******************************************
CAPTURE_BUFFER_WIDTH        equ 1024        ;视频俘获缓存的宽度
CAPTURE_BUFFER_HEIGHT       equ 512         ;和高度

ifdef DOSLIB
EXT_MEM_BFR_SIZE            equ 2048        ;扩充内存缓存的大小
endif

; System metrics                            ;系统量度
SM_VIDEOWIDTH       equ     0               ;视频宽度
SM_VIDEOHEIGHT      equ     1               ;高度
SM_BOARDTYPE        equ     2               ;板型
SM_VERSION equ      3                       ;版本号
SM_INTERLACE        equ     4               ;隔行扫描
SM_REPLICATE        equ     5               ;场复制
SM_IMAGEWIDTH       equ     6               ;图象宽度
SM_IMAGEHEIGHT      equ     7               ;高度
SM_IMAGETYPE        equ     8               ;类型

; Mode flags                                ;模式旗
MF_PACKED           equ     40h             ;象数组
MF_PLANAR           equ     00h             ;彩色位面
MF_INTERLACED       equ     02h             ;隔行扫描的
MF_PLUSPOLARITY     equ     08h             ;多极性

; Video board types                         ;视频板的类型
```

```
BT_YUV411           equ    0       ;本书介绍的 GRANDVIDEO 卡就是这种
BT_RGB16            equ    1       ;红绿蓝 16 色
BT_YUV211           equ    2       ;另二种 YUV 格式
BT_YUV422           equ    3
BT_RGB24            equ    4       ;红绿蓝 24 色

; Bitmap types                     ;图象的位图格式
BM_DIB24            equ    0       ;Windows DIB 24 bpp 真彩色
BM_DIB8P            equ    1       ;Windows DIB 8 bpp 被调色
BM_DIB8G            equ    2       ;Windows DIB 8 bpp 灰度
BM_DIB4D            equ    3       ;Windows DIB 4 bpp 抖动色
BM_TRG32            equ    4       ;Targa 32 bpp 真彩色
BM_TRG24            equ    5       ;Targa 24 bpp 真彩色
BM_TRG16            equ    6       ;Targa 16 bpp 真彩色
BM_YUV411           equ    7       ;IBM MMotion 格式 YUV
BM_TIFF8P           equ    8       ;TIFF 8 bit palettized
BM_TIFF8G           equ    9       ;TIFF 8 bit gray—scale
BM_TIFF4D           equ    10      ;TIFF 4 bit dithered
BM_PCX8P            equ    11      ;PCX 8 bit palettized
BM_PCX8G            equ    12      ;PCX 8 bit gray—scale
BM_PCX4D            equ    13      ;PCX 4 bit dithered
BM_GIF8P            equ    14      ;GIF 8 bit palettized
BM_GIF8G            equ    15      ;GIF 8 bit gray—scale
BM_JPEG             equ    16      ;JPEG 压缩 4:2:2 YUV

LOCLK               equ    0       ;时钟低位
HICLK               equ    1       ;时钟高位
ADR9051             equ    8AH     ;数字多标准译码器
ADR4680             equ    88H     ;视频控制组件
ADR7192             equ    0E0H    ;Digital color space converter
ADR7191             equ    8AH     ;Square pixel digital multistandard decoder
ADRDMSD             equ    8AH     ;Digital multistandard decoder
I2C_ADR             equ    0       ;I2C 地址
I2C_SUBADR          equ    1
I2C_DAT             equ    2       ;I2C 数据
I2C_SCL             equ    0       ;I2C 串行时钟
I2C_SDA             equ    1       ;I2C 串行数据

BRIGHTNESS          equ    0       ;亮度

NO_COLOR_SETTINGS   equ    8       ;可用的彩色控制数
NO_COLOR_CONTROLS   equ    7       ;实际彩色控制数
```

```
NO_SKEW_FACTORS         equu    7       ;偏移因子(skew factors)数
NO_VIDEO_MODES          equ     16      ;可支持的视频模式数
NO_PHIXEL_CHIPS         equ     4       ;所支持的 Phixel 芯片数

LUMA_BITMASK            equ     07H     ;灰度屏蔽寄存器
CROMA_BITMASK           equ     08H     ;色度屏蔽寄存器
INTERRUPT_POLL          equ     09H     ;中断
VIDEO_VSYNC             equ     04H     ;视频垂直同步
VGA_VSYNC               equ     10H     ;VGA 垂直同步
VGA_HSYNC               equ     20H     ;VGA 水平同步
PLL_DIVISOR_INDEX       equ     13H     ;锁相环分频系数指针
PLL_DIVISOR_DATA        equ     12H     ;数据
PLL_DIVISOR_LO          equ     0       ;低位
PLL_DIVISOR_HI          equ     1       ;高位
I2C_BUS_CONTROL         equ     18H     ;I2C 总线控制
AV_MODE_CNTL            equ     20H     ;视频采集模式控制寄存器
AW_MODE_CNTL            equ     21H     ;采集窗模式控制寄存器
AW_X_START_LO           equ     22H     ;X 开始低位
AW_X_START_HI           equ     23H     ;高位
AW_Y_START_LO           equ     24H     ;Y 开始低位
AW_Y_START_HI           equ     25H     ;高位
AW_X_END_LO             equ     26H     ;X 结束低位
AW_X_END_HI             equ     27H     ;高位
AW_Y_END_LO             equ     28H     ;Y 结束低位
AW_Y_END_HI             equ     29H     ;高位
AB_ADDR_LO              equ     2AH     ;采集后在帧缓存中写的地址低位
AB_ADDR_MI              equ     2BH     ;中间位
AB_ADDR_HI              equ     2CH     ;高位
AW_X_SCALING            equ     2DH     ;输入水平压缩
AW_Y_SCALING            equ     2EH     ;垂直压缩
AW_Y_SCALING_ODD        equ     2FH     ;奇数行专用
SCALE_CONTROL           equ     38H     ;标定控制寄存器
DW_MODE_CNTL            equ     40H     ;显示窗模式控制寄存器
DW_X_START_LO           equ     41H     ;显示窗 X 开始低寄存器
DW_X_START_HI           equ     42H     ;高
DW_Y_START_LO           equ     43H     ;Y 低
DW_Y_START_HI           equ     44H     ;Y 高
DW_X_END_LO             equ     45H     ;显示窗 X 结束低
DW_X_END_HI             equ     46H     ;高
DW_Y_END_LO             equ     47H     ;Y 低
DW_Y_END_HI             equ     48H     ;Y 高
DW_X_PANNING_LO         equ     49H     ;X 平移低位寄存器
```

```
DW_Y_PANNING_LO       equ    4AH            ;Y
DW_PANNING_HI         equ    4BH            ;平移高位共用寄存器
SHIFT_CLK_START       equ    4CH            ;乙位时钟开始位置
DW_WINDOW_ZOOM        equ    4DH            ;显式窗放大寄存器
TRANS_COLOR           equ    4EH            ;透明彩色寄存器
INTERLACE_CNTL        equ    50H            ;显式隔行扫描输出控制
INTERLACE_ON          equ    01H            ;允许隔行扫描输出
REPLICATE_ON          equ    08H            ;允许场复制
NON_MULTIPLEXED       equ    10H            ;视频输入不准多路

FRAME_BUF_WIDTH       equ    2048           ;采集缓存宽度是 2K
NSELECTORS            equ    16             ;1MB 视频缓存的扇区数

ifdef DOSLIB                                ;DOS 库存在的话
DOS_SERVICE           equ    21H            ;DOS 服务请求中断号
DOS_CREATE_FILE       equ    3CH            ;DOS 服务:建文件
DOS_OPEN_FILE         equ    3DH            ;DOS 打开文件
DOS_CLOSE_FILE        equ    3EH            ;DOS 关闭文件
DOS_READ_FILE         equ    3FH            ;DOS 读文件
DOS_WRITE_FILE        equ    40H            ;DOS 写文件
endif

; Configuration flags                       ;配置旗
CF_PAL                equ    0              ;输入制式是 PAL
CF_NTSC               equ    1              ;NTSC
CF_HASPLL             equ    2              ;板内有锁相环
CF_REPLICATE          equ    REPLICATE_ON   ;允许场复制
                      ;以下是用户定义结构
LONG        struc                           ;长整数
lo          dw       ?                      ;低字
hi          dw       ?                      ;高字
LONG        ends

FARPOINTER            struc                 ;远指针
off         dw       ?                      ;偏移地址
sel         dw       ?                      ;断地址
FARPOINTER            ends

REGENTRY struc                              ;寄存器入口
rIndex                db ?                  ;指针
rData                 db ?                  ;数据
REGENTRY ends
```

```
PCVIDEOTBL struc                                        ;PCVIDEO 表格
cPCVideoRegs           dw ?                             ;寄存器总数
aPCVideoRegs           db 64 * SIZE REGENTRY dup (?)    ;寄存器指针和数据
PCVIDEOTBL ends

PHIXELCHIP struc                                        ;PHIXEL 芯片参数结构
aChipId                db 10 dup(?)                     ;I2C 芯片确认
wPhixelAddr            dw ?                             ;I2C 地址
cPhixelRegs            dw ?                             ;寄存器计数
aPhixelRegs            db 32 * SIZE REGENTRY dup (?)    ;寄存企器指针和数据
PHIXELCHIP ends

PHIXELTBL struc                                         ;PHIXEL 表结构
cPhixelChips           dw ?
aPhixelChips           db NO_PHIXEL_CHIPS * SIZE PHIXELCHIP dup (?)
PHIXELTBL ends

MODE                   struc                            ;显式模式参数结构
DispWinSkewX           dw ?                             ;加到显示窗寄存器的 X 偏移量
DispWinSkewY           dw ?                             ;Y
DispAddrSkewX          dw 4 dup (?)                     ;地址的 X 偏移量
DispAddrSkewY          dw 4 dup (?)                     ;Y
ShiftClkStart          dw 4 dup (?)                     ;移位时钟开始位置
PaletteSkew            dw ?                             ;2 MSBs of 显示窗控制寄存器
VGAWidth               dw ?                             ;VGA 分辨率
VGAHeight              dw ?                             ;高度
PLLDivisor             dw ?                             ;锁相环的分频系数
fModeFlags             dw ?                             ;杂项旗
MODE                   ends

CFGSTRUCT              struc                            ;配置文件结构
wPortAddr              dw ?                             ;口地址
wVideoAddr             dw ?                             ;视频地址
wCfgFlags              dw ?                             ;配置旗
wVideoSource           dw ?                             ;视频源
bColorSettings   db NO_COLOR_SETTINGS dup (?)                 ;设定彩色
PCVideoTable     db SIZE PCVIDEOTBL dup (?)                   ;PCVideo 表格
PhixelTable      db SIZE PHIXELTBL dup (?)                    ;Phixel 表格
aMode            db SIZE MODE * NO_VIDEO_MODES dup (?) ;各种可用显示模示参数表
wNTSCAcqWindowdw 4 dup (?)                                    ;NTSC 采集窗口
wPALAcqWindow dw 4 dup (?)                                    ;PLL 采集窗口
aReserved        db 44 dup (?) ;保留
CFGSTRUCT        ends
```

```
? PLM=1                                           ;采用 Pascal 调用习惯,即子程序
                                                  ;最左边的参数首先压栈

ifdef DOSLIB
? WIN=0                                           ;无 Windows 的收尾/开场程序
else
? WIN=1                                           ;采用 Windows 的收尾/开场程序
endif

ifdef DOSLIB
GDTEntry struc                                    ;请求存取括充内存
GDTEntry_Size dw ?                                ;段大小
GDTEntry_Address db 3 dup (?)                     ;24 位开始地址
GDTEntry_Access db ?                              ;存取旗
GDTEntry_Reserved dw ?
GDTEntry ends

GDT struc                                         ;全局表
GDT_Dummy0 db 16 dup (?)                          ;哑指针
GDT_Entry1 db SIZE GDTEntry dup (?);入口 1
GDT_Entry2 db SIZE GDTEntry dup (?);入口 2
GDT_Dummy1 db 16 dup (?)
GDT ends
endif

; 定义输出程序的宏。DOS 程序定义为近,Windows 输出程序定义为远

ExportProc macro name, regs
if ? WIN
cProc name, <FAR,PUBLIC>, <regs>
else
cProc name, <NEAR,PUBLIC>, <regs>
endif
endm

externEP macro procname
if ? WIN
externFP procname
else
externNP procname
endif
endm
```

以下两个C语言的头文件，它的内容大多数与前述PCVIDEO.INC重复，故不再加中文注释。

```
/*****************************************************************************
* 程序名:      PCVIDEO.DLL                                                   *
* 模块  :      PCVIDEO.H                                                     *
* 描述  :      'C'编程协议定义                                                *
* 版本  :      1.6                                                           *
* 版权  (C) Chips and Technologies, Inc. 1991, 1992                          *
*****************************************************************************/
#include <typedef.h>
int EXPORT PASCAL PCV_CreateWindow(WORD, WORD, WORD, WORD, BOOL);
int EXPORT PASCAL PCV_SetWindowPosition(WORD, WORD);
int EXPORT PASCAL PCV_SetWindowSize(WORD, WORD, BOOL);
int EXPORT PASCAL PCV_SetVideoScaling(WORD,WORD,BOOL);
int EXPORT PASCAL PCV_SetCaptureAddress(WORD,WORD);
int EXPORT PASCAL PCV_SetDisplayWindow(WORD,WORD,WORD,WORD);
int EXPORT PASCAL PCV_SetAcquisitionWindow(WORD,WORD,WORD,WORD);
int EXPORT PASCAL PCV_PanWindow(WORD, WORD);
int EXPORT PASCAL PCV_SetColorKey(WORD);
int EXPORT PASCAL PCV_FreezeVideo(void);
int EXPORT PASCAL PCV_UnFreezeVideo(void);
int EXPORT PASCAL PCV_SetWriteProtectMask(WORD);
int EXPORT PASCAL PCV_EnableVideo(void);
int EXPORT PASCAL PCV_DisableVideo(void);
int EXPORT PASCAL PCV_SaveImageRect (LPSTR, WORD, WORD, WORD, WORD, WORD,
WORD);
int EXPORT PASCAL PCV_LoadImageRect(LPSTR, WORD, WORD);
int EXPORT PASCAL PCV_ReadImageRect(PSTR, WORD, WORD, WORD, WORD, WORD, char
far *);
int EXPORT PASCAL PCV_WriteImageRect (PSTR, PSTR, WORD, WORD, WORD, WORD,
WORD);
int EXPORT PASCAL PCV_GetInputFormat(void);
int EXPORT PASCAL PCV_SetInputFormat(WORD);
int EXPORT PASCAL PCV_SetColor(WORD, BYTE);
BYTE EXPORT PASCAL PCV_GetColor(WORD);
void EXPORT PASCAL PCV_ResetColors(void);
int EXPORT PASCAL PCV_ClearVideoRect(WORD, WORD, WORD, WORD);
int EXPORT PASCAL PCV_LoadConfiguration(void);
int EXPORT PASCAL PCV_SaveConfiguration(void);
int EXPORT PASCAL PCV_GetPortAddress(void);
int EXPORT PASCAL PCV_SetPortAddress(int);
int EXPORT PASCAL PCV_GetVideoAddress(void);
int EXPORT PASCAL PCV_SetVideoAddress(int);
int EXPORT PASCAL PCV_GetVideoSource(void);
```

```
int EXPORT PASCAL PCV_SetVideoSource(int);
WORD EXPORT PASCAL PCV_GetSkewFactor(int);
int EXPORT PASCAL PCV_SetSkewFactor(int, WORD);
void EXPORT PASCAL PCV_ResetSkewFactors(void);
WORD EXPORT PASCAL PCV_GetSystemMetrics(WORD);
void EXPORT PASCAL PCV_EnableInterlace(void);
void EXPORT PASCAL PCV_DisableInterlace(void);
void EXPORT PASCAL PCV_HorizontalZoom(int);
void EXPORT PASCAL PCV_VerticalZoom(int);
void EXPORT PASCAL PCV_EnableFieldReplication(void);
void EXPORT PASCAL PCV_DisableFieldReplication(void);
void EXPORT PASCAL PCV_SetRegister(WORD, BYTE);
BYTE EXPORT PASCAL PCV_GetRegister(WORD);
void EXPORT PASCAL PCV_TurnBorder(WORD);
int EXPORT PASCAL PCV_Initialize(void);
int EXPORT PASCAL PCV_Exit(void);
void EXPORT PASCAL PCV_WaitVGARetrace(void);
void EXPORT PASCAL PCV_EnableColorKey(void);
void EXPORT PASCAL PCV_DisableColorKey(void);
WORD EXPORT PASCAL PCV_GetProfileInt(LPSTR, LPSTR);
int EXPORT PASCAL PCV_GetProfileString(LPSTR,LPSTR,LPSTR,int);

#define CAPTURE_BUFFER_WIDTH    1024    /* Dimensions of the capture */
#define CAPTURE_BUFFER_HEIGHT   512     /* buffer in pixels */

#define LOCLK           0
#define HICLK           1

#define PLL_DIVISOR_INDEX   0x13
#define PLL_DIVISOR_DATA    0x12
#define PLL_DIVISOR_LO      0
#define PLL_DIVISOR_HI      1
#define AV_MODE_CNTL        0x20    /* Acquisition video mode register */
#define AW_MODE_CNTL        0x21    /* Acquisition window control register */
#define NON_MULTIPLEXED 0x10        /* Input is not multiplexed. */
#define ACQUIRE_FIELD       4
#define SHIFT_CLK_START     0x4C
#define DW_MODE_CNTL        0x40

#define INTERLACE_CNTL      0x50
#define INTERLACE_ON        1
#define REPLICATE_ON        8
```

```
#define I2C_ADR            0
#define I2C_SUBADR         1
#define I2C_DAT            2

#define I2C_SCL            0
#define I2C_SDA            1

/* Color adjustments */
#define BRIGHTNESS         0
#define SATURATION         1
#define CONTRAST           2
#define HUE                3
#define RED                4
#define GREEN              5
#define BLUE               6

/* Phixel chip addresses */
#define ADRDMSD          0x8A    /* Digital multistandard decoder */
#define ADR7191          0x8A    /* Square pixel digital multistandard decoder 4:2:2 */
#define ADR7192          0xE0    /* Color space converter */
#define ADR9051          0x8A    /* Digital multistandard decoder 4:1:1 */
#define ADR4680          0x88    /* Color processor */

#define HUE_CONTROL          7

/* System metrics */
#define SM_VIDEOWIDTH        0
#define SM_VIDEOHEIGHT       1
#define SM_BOARDTYPE         2
#define SM_VERSION           3
#define SM_INTERLACE         4
#define SM_REPLICATE         5
#define SM_IMAGEWIDTH        6
#define SM_IMAGEHEIGHT       7
#define SM_IMAGETYPE         8

/* Mode flags */
#define MF_PACKED            0x40
#define MF_PLANAR            0x00
#define MF_INTERLACED        0x02
#define MF_PLUSPOLARITY      0x08

/* Video board types */
```

```
#define BT_YUV411     0
#define BT_RGB16      1
#define BT_YUV211     2
#define BT_YUV422     3
#define BT_RGB24      4

/* Bitmap types */
#define BM_DIB24      0    /* Windows DIB 24 bit true color */
#define BM_DIB8P      1    /* Windows DIB 8 bit palettized */
#define BM_DIB8G      2    /* Windows DIB 8 bit gray-scale */
#define BM_DIB4D      3    /* Windows DIB 4 bit dithered */
#define BM_TRG32      4    /* Targa 32 bit true color */
#define BM_TRG24      5    /* Targa 24 bit true color */
#define BM_TRG16      6    /* Targa 16 bit true color */
#define BM_YUV411     7    /* IBM MMotion format 4:1:1 YUV */
#define BM_TIFF8P     8    /* TIFF 8 bit palettized */
#define BM_TIFF8G     9    /* TIFF 8 bit gray-scale */
#define BM_TIFF4D     10   /* TIFF 4 bit dithered */
#define BM_PCX8P      11   /* PCX 8 bit palettized */
#define BM_PCX8G      12   /* PCX 8 bit gray-scale */
#define BM_PCX4D      13   /* PCX 4 bit dithered */
#define BM_GIF8P      14   /* GIF 8 bit palettized */
#define BM_GIF8G      15   /* GIF 8 bit gray-scale */
#define BM_JPEG       16   /* JPEG compressed 4:2:2 YUV */

#define MMP_HDR_STRING    "YUV12C"   /* Header string for MMotion file */
#define BMP_HDR_STRING    "BM"       /* Header string for Windows DIB */

typedef struct MMPHDR {
        cha     mhIDString[6];
        WORD    mhReserved[3];
        WORD    mhWidth;
        WORD    mhHeight;
        } MMPHDR;
typedef MMPHDR FAR *LPMMPHDR;

typedef struct TRGHDR {
        BYTE        bIDFieldSize;         /* Characters in ID field */
        BYTE        bClrMapType;          /* Color map type */
        BYTE        bImageType;           /* Image type */
        BYTE        lClrMapSpec[5];       /* Color map specification */
        WORD        wXOrigin;             /* X origin */
        WORD        wYOrigin;             /* Y origin */
```

```
    WORD      wWidth;                  /* Bitmap width */
    WORD      wHeight;                 /* Bitmap height */
    BYTE      bBitsPixel;              /* Bits per pixel */
    BYTE      bImageDescriptor;        /* Image descriptor */
    } TRGHDR;
typedef TRGHDR FAR *LPTRGHDR;

#define ORIGIN_TOP_LEFT 0x20           /* Origin is at top left. */

#define NO_COLOR_SETTINGS 8            /* Available color controls */
#define NO_COLOR_CONTROLS 7            /* Actual number of color controls */
#define NO_SKEW_FACTORS 7              /* Number of skew factors */
#define NO_VIDEO_MODES 16              /* Number of video modes supported */
#define NO_PHIXEL_CHIPS 4              /* Number of Phixel chips supported */

/* Configuration flags */
#define CF_PAL              0              /* Input format is PAL */
#define CF_NTSC             1              /* Input format is NTSC */
#define CF_HASPLL           2              /* Board has a phase-locked loop */
#define CF_REPLICATE        REPLICATE_ON   /* Replicate field feature enabled */

/* Skew adjustment IDs */
#define SF_DISPWINSKEWX       0        /* Display Window X Skew */
#define SF_DISPWINSKEWY       1        /* Display Window Y Skew */
#define SF_DISPADDRSKEWX      2        /* Display Address X Skew */
#define SF_DISPADDRSKEWY      3        /* Display Address Y Skew */
#define SF_SHIFTCLOCKSTART    4        /* Shift Clock Start */
#define SF_PALETTESKEW        5        /* Palette Skew */
#define SF_PLLDIVISOR         6        /* Phase Lock Loop Divisor */

/* Skew adjustment ranges */
#define DISPWINSKEWXRANGE       0x7FF    /* Display Window X Skew Range */
#define DISPWINSKEWYRANGE       0x3FF    /* Display Window Y Skew Range */
#define DISPADDRSKEWXRANGE      0x3FF    /* Display Address X Skew Range */
#define DISPADDRSKEWYRANGE      0x1FF    /* Display Address Y Skew Range */
#define SHIFTCLOCKSTARTRANGE    0x7F     /* Shift Clock Start Range */
#define PALETTESKEWRANGE        0x03     /* Palette Skew Range */
#define PLLDIVISORRANGE         0x3FF    /* Phase Lock Loop Divisor Range */

#define MAX_PALETTE_ENTRIES     236      /* Windows reserves 20 of the 256 colors */
#define PF_PALFILE              0x4000   /* Output a palette file */

typedef struct REGENTRY {
```

```
    BYTE            rIndex;                     /* Register index */
    BYTE            rData;                      /* Register data */
    } REGENTRY;

typedef struct PCVIDEOTBL {
    WORD            cPCVideoRegs;               /* Count of registers */
    REGENTRY        aPCVideoRegs[64];           /* Register index and data */
    } PCVIDEOTBL;

typedef struct PHIXELCHIP {
    char            aChipID[10];                /* i2c chip id */
    WORD            wPhixelAddr;                /* i2c address of chip */
    WORD            cPhixelRegs;                /* Count of registers */
    REGENTRY        aPhixelRegs[32];            /* Register index and data */
    } PHIXELCHIP;

typedef struct PHIXELTBL {
    WORD            cPhixelChips;
    PHIXELCHIP      aPhixelChips[NO_PHIXEL_CHIPS];
    } PHIXELTBL;

typedef struct MODE {
    WORD            DispWinSkewX;               /* Offset to add to display window registers */
    WORD            DispWinSkewY;
    WORD            DispAddrSkewX[4];           /* Offsets to be added to display address */
    WORD            DispAddrSkewY[4];
    WORD            ShiftClkStart[4];           /* Shift clock start position */
    WORD            PaletteSkew;                /* 2 MSBs of Display Window Control Reg. */
    WORD            VGAWidth;                   /* VGA resolution */
    WORD            VGAHeight;
    WORD            PLLDivisor;                 /* Divisor for the PLL */
    WORD            fModeFlags;                 /* Miscellaneous flags */
} MODE;
typedef MODE *PMODE;

typedef struct CFGSTRUCT {
    WORD            wPortAddr;
    WORD            wVideoAddr;
    WORD            wCfgFlags;

    WORD            wVideoSource;
    BYTE            bColorSettings[NO_COLOR_SETTINGS];
    PCVIDEOTBL      PCVideoTable;
```

```
    PHIXELTBL        PhixelTable;
    MODE             aMode[NO_VIDEO_MODES];
    WORD             wNSTCAcqWindow[4];
    WORD             wPALAcqWindow[4];
    BYTE             aReserved[44];
} CFGSTRUCT;

typedef unsigned int uint;
typedef unsigned long ulong;

/*****************************************************************
 * Program:     PCVIDEO.DLL                                      *
 * Module:      TYPEDEF.H                                        *
 * Description:        Windows 协议定义.                         *
 * Version:    1.6                                               *
 * Author:    Olive Tao                                          *
 * Copyright (C) Chips and Technologies, Inc. 1991, 1992         *
 *****************************************************************/

#define TRUE                 1
#define FALSE                0

#ifdef DOSLIB
#define FAR
#define EXPORT               near
#define ADDRESS              WORD
#else
#define FAR                  far
#define EXPORT               far
#define ADDRESS              DWORD
#endif

#define NEAR                 near
#define LONG                 long
#define VOID                 void
#define PASCAL               pascal

#define MAKELONG(a, b)       ((LONG)(((WORD)(a)) | ((DWORD)((WORD)(b))) << 16))
#define LOWORD(l)            ((WORD)(l))
#define HIWORD(l)            ((WORD)(((DWORD)(l) >> 16) & 0xFFFF))
#define LOBYTE(w)            ((BYTE)(w))
#define HIBYTE(w)            ((BYTE)(((WORD)(w) >> 8) & 0xFF))
```

```
typedef int                    BOOL;
typedef unsigned char          BYTE;
typedef unsigned int           WORD;
typedef unsigned long          DWORD;
typedef char near              *PSTR;
typedef char near              *NPSTR;
typedef char FAR               *LPSTR;
typedef BYTE near              *PBYTE;
typedef BYTE FAR               *LPBYTE;
typedef int near               *PINT;
typedef int FAR                *LPINT;
typedef WORD near              *PWORD;
typedef WORD FAR               *LPWORD;
typedef long near              *PLONG;
typedef long FAR               *LPLONG;
typedef DWORD near             *PDWORD;
typedef DWORD FAR              *LPDWORD;
typedef void FAR               *LPVOID;
typedef WORD                   HANDLE;
typedef HANDLE                 HWND;
typedef HANDLE                 *PHANDLE;
typedef HANDLE NEAR            *SPHANDLE;
typedef HANDLE FAR             *LPHANDLE;
typedef HANDLE                 GLOBALHANDLE;
typedef HANDLE                 LOCALHANDLE;
typedef int (FAR PASCAL  *FARPROC)();
typedef int (NEAR PASCAL  *NEARPROC)();
typedef HANDLE                 HSTR;
typedef HANDLE                 HICON;
typedef HANDLE                 HDC;
typedef HANDLE                 HMENU;
typedef HANDLE                 HPEN;
typedef HANDLE                 HFONT;
typedef HANDLE                 HBRUSH;
typedef HANDLE                 HBITMAP;
typedef HANDLE                 HCURSOR;
typedef HANDLE                 HRGN;
typedef HANDLE                 HPALETTE;
typedef DWORD                  COLORREF;

#ifdef DOSLIB
typedef struct tagRECT
{
```

```
    int      left;
    int      top;
    int      right;
    int      bottom;
} RECT;

typedef RECT                 * PRECT;
typedef RECT NEAR            * NPRECT;
typedef RECT FAR             * LPRECT;

typedef struct tagPOINT
{
    int      x;
    int      y;
} POINT;

typedef POINT                * PPOINT;
typedef POINT NEAR           * NPPOINT;
typedef POINT FAR            * LPPOINT;

/ * Bitmap Header Definition * /
typedef struct tagBITMAP
{
    int          bmType;
    int          bmWidth;
    int          bmHeight;
    int          bmWidthBytes;
    BYTE         bmPlanes;
    BYTE         bmBitsPixel;
    LPSTR        bmBits;
} BITMAP;

typedef BITMAP               * PBITMAP;
typedef BITMAP NEAR          * NPBITMAP;
typedef BITMAP FAR           * LPBITMAP;

typedef struct tagRGBTRIPLE {
        BYTE rgbtBlue;
        BYTE rgbtGreen;
        BYTE rgbtRed;
} RGBTRIPLE;

typedef struct tagRGBQUAD {
```

```
    BYTE rgbBlue;
    BYTE rgbGreen;
    BYTE rgbRed;
    BYTE rgbReserved;
} RGBQUAD;

/* structures for defining DIBs */
typedef struct tagBITMAPCOREHEADER {
    DWORD     bcSize;          /* used to get to color table */
    WORD      bcWidth;
    WORD      bcHeight;
    WORD      bcPlanes;
    WORD      bcBitCount;
} BITMAPCOREHEADER;

typedef BITMAPCOREHEADER FAR *LPBITMAPCOREHEADER;
typedef BITMAPCOREHEADER *PBITMAPCOREHEADER;

typedef struct tagBITMAPINFOHEADER{
    DWORD     biSize;
    DWORD     biWidth;
    DWORD     biHeight;
    WORD      biPlanes;
    WORD      biBitCount;
    DWORD     biCompression;
    DWORD     biSizeImage;
    DWORD     biXPelsPerMeter;
    DWORD     biYPelsPerMeter;
    DWORD     biClrUsed;
    DWORD     biClrImportant;
} BITMAPINFOHEADER;

typedef BITMAPINFOHEADER FAR *LPBITMAPINFOHEADER;
typedef BITMAPINFOHEADER *PBITMAPINFOHEADER;

typedef struct tagBITMAPINFO {
  BITMAPINFOHEADER      bmiHeader;
  RGBQUAD               bmiColors[1];
} BITMAPINFO;

typedef BITMAPINFO FAR *LPBITMAPINFO;
typedef BITMAPINFO *PBITMAPINFO;
```

```
typedef struct tagBITMAPCOREINFO {
    BITMAPCOREHEADER     bmciHeader;
    RGBTRIPLE            bmciColors[1];
} BITMAPCOREINFO;

typedef BITMAPCOREINFO FAR *LPBITMAPCOREINFO;
typedef BITMAPCOREINFO *PBITMAPCOREINFO;

typedef struct tagBITMAPFILEHEADER {
        WORD     bfType;
        DWORD    bfSize;
        WORD     bfReserved1;
        WORD     bfReserved2;
        DWORD    bfOffBits;
} BITMAPFILEHEADER;

typedef BITMAPFILEHEADER FAR *LPBITMAPFILEHEADER;
typedef BITMAPFILEHEADER *PBITMAPFILEHEADER;

/* constants for the biCompression field */
#define BI_RGB       0L
#define BI_RLE8      1L
#define BI_RLE4      2L

/* Flags specifying the access mode for files. */
#define P_MODE                  S_IREAD | S_IWRITE
#define F_MODE                  O_RDWR | O_CREAT | O_BINARY
#define F_READ                  O_RDONLY | O_BINARY
#define F_WRITE                 O_WRONLY | O_BINARY

#define filecreat(a, b)         open(a, b, P_MODE)
#define fileread(a, b, c)       read(a, b, c)
#define filewrite(a, b, c)      write(a, b, c)
#define fileopen(a, b)          open(a, b)
#define fileclose(a)            close(a)
#define stringf                 sprintf
#define stringcmp               stricmp
#define stringcpy               strcpy
#define stringlen               strlen
#else          /* Windows 3.0 */

/* Global memory flags */
#define GMEM_FIXED              0x0000
```

```
#define GMEM_MOVEABLE           0x0002
#define GMEM_NOCOMPACT          0x0010
#define GMEM_NODISCARD          0x0020
#define GMEM_ZEROINIT           0x0040
#define GMEM_MODIFY             0x0080
#define GMEM_DISCARDABLE        0x0100
#define GMEM_NOT_BANKED         0x1000
#define GMEM_SHARE              0x2000
#define GMEM_DDESHARE           0x2000
#define GMEM_NOTIFY             0x4000
#define GMEM_LOWER              GMEM_NOT_BANKED

#define SEEK_SET                0

HANDLE FAR PASCAL GlobalAlloc(WORD, DWORD);
LPSTR FAR PASCAL GlobalLock(HANDLE);
BOOL FAR PASCAL GlobalUnlock(HANDLE);
HANDLE FAR PASCAL GlobalFree(HANDLE);

/* OpenFile() Flags */
#define OF_READ                 0x0000
#define OF_WRITE                0x0001
#define OF_READWRITE            0x0002
#define OF_SHARE_COMPAT         0x0000
#define OF_SHARE_EXCLUSIVE      0x0010
#define OF_SHARE_DENY_WRITE     0x0020
#define OF_SHARE_DENY_READ      0x0030
#define OF_SHARE_DENY_NONE      0x0040
#define OF_PARSE                0x0100
#define OF_DELETE               0x0200
#define OF_VERIFY               0x0400
#define OF_CANCEL               0x0800
#define OF_CREATE               0x1000
#define OF_PROMPT               0x2000
#define OF_EXIST                0x4000
#define OF_REOPEN               0x8000

/* Flags specifying the access mode for files. */
#define F_MODE                  0
#define F_READ                  OF_READ
#define F_WRITE                 OF_WRITE

int     FAR cdecl wsprintf(LPSTR,LPSTR,...);
```

```
int      FAR PASCAL lopen( LPSTR, int );
int      FAR PASCAL lclose( int );
int      FAR PASCAL lcreat( LPSTR, int );
LONG FAR PASCAL llseek( int, long, int );
WORDFAR PASCAL lread( int, LPSTR, int );
WORDFAR PASCAL lwrite( int, LPSTR, int );
WORDFAR PASCAL GetSystemDirectory(LPSTR,WORD);

#define filecreat(a, b)            lcreat(a, b)
#define fileread(a, b, c)          _lread(a, b, c)
#define filewrite(a, b, c)         _lwrite(a, b, c)
#define fileopen(a, b)             _lopen(a, b)
#define fileclose(a)               _lclose(a)
#define stringf                    wsprintf
#define stringcmp                  lstrcmpi
#define stringcpy                  lstrcpy
#define stringlen                  lstrlen
#endif                             /* Windows 3.0 */

/*************************************************
 * Macro definitions for 与 WINDOWS 动态联结库和 DOS 库接口的宏定义.    *
 ************************************************ /

/* Macros min(a,b) and max(a,b) */
#define min(a,b)         (((a) < (b)) ? (a) : (b))
#define max(a,b)         (((a) > (b)) ? (a) : (b))
```

以下的 PCVIDEO. INI 文件定义了 PCVIDEO 初始化后的各种寄存器初始设定值。

此文件不能加注解，否则将影响 PCVIDEO 软件的正确执行。但是为了方便读者对 PCVIDEO 源程序的理解，我们一律加中文注释。当读者需要从源程序编译联接成实用软件时，还请用原盘所给的 PCVIDEO. INI 文件。此外还建议读者把它和有关硬件知识对照阅读。当读者购买了一块 GRAND VIDEO 卡而不知它的档次时，查随卡软盘中的 PCVIDEO. INI 文件可以了解。

PCVIDEO. INI 文件

```
[PCVIDEO]
BufferType=YUV411          ;缓存类型 YUV411
PLL=No                     ;板中无锁相环
PortAddr=0xAD6             ;视频窗控制器 82C9001A 的口地址是 0AD6(16 进制)
VideoAddr=0x09             ;视频缓存地址是 9 (8 兆字节以外)
VideoFormat=NTSC           ;视频制式是 NTSC
VideoSource=0              ;选通第一视频源
RegCount=43                ;82C9001A 共 43 个寄存器
```

```
Reg1=0x10               ;口地址的高位 0A,隔行扫描
Reg6=0x19               ;
Reg7=0xFF
Reg8=0xFF               ;场复制
Reg9=0x00               ;中断屏蔽控制
Reg12=0x26              ;
Reg13=0x10
Reg18=0x33
Reg20=0x21              ;视频采集控制
Reg21=0x0D              ;采集窗控制
Reg22=0x10              ;X 开始值=16
Reg23=0x00
Reg24=0x20              ;Y =32
Reg25=0x00
Reg26=0xBC              ;X 结束值=700
Reg27=0x02
Reg28=0xEC              ;Y 结束值=492
Reg29=0x01
Reg2A=0x00              ;视频缓存开始地址=0
Reg2B=0x00
Reg2C=0x00
Reg2D=0x20              ;X 标定
Reg2E=0x20              ;Y
Reg2F=0x20              ;Y 奇数行标定
Reg30=0x06              ;
Reg38=0x94
Reg40=0xA3              ;显示窗控制
Reg41=0x00              ;X 开始值=0
Reg42=0x00
Reg43=0x20              ;Y =32
Reg44=0x00
Reg45=0x00              ;X 结束=0
Reg46=0x00
Reg47=0x00              ;Y 结束=0
Reg48=0x00
Reg49=0xF8              ;X 平移
Reg4A=0xE0              ;Y 平移
Reg4B=0x11              ;
Reg4C=0x12              ;移位开始
Reg4D=0x00              ;放大
Reg4E=0x2D              ;
Reg4F=0x00
Reg50=0x00
```

```
[I2CChips]                  ;I2C 总线上芯片
ChipsCount=                 ;共二片
Chip1=SAA9051               ;
Chip2=TDA4680

[SAA9051]
I2CAddr=0x8A
RegCount=12
Reg0=0x64
Reg1=0x35
Reg2=0x0A
Reg3=0xF8
Reg4=0xCD
Reg5=0xFE
Reg6=0x22
Reg7=0x00
Reg8=0x77
Reg9=0xE0
RegA=0x42
RegB=0x00

[TDA4680]
I2CAddr=0x88
RegCount=16
Reg0=0x3F
Reg1=0x12
Reg2=0x16
Reg3=0x22
Reg4=0x20
Reg5=0x20
Reg6=0x20
Reg7=0x20
Reg8=0x20
Reg9=0x20
RegA=0x3F
RegB=0x00
RegC=0x89
RegD=0x10
RegE=0x00
RegF=0x00

[ModeInfo]                  ;参看 PCVIDEO. INC 或 PCVIDEO. H 中有关显示模式的结构定义
```

```
ModeCount=16
Mode0=0x2E 0x20 0 0 0 0 0x20 0x0F 0 0 0x12 0 0 0 0x01 0x280 0x1E0 0xA317 0
Mode1=0x8D 0x1A 0x30 0x12 0 0 0x1A 0x0D 0 0 0x12 0x14 0 0 0x01 0x2D0 0x21C 0xA330 0
Mode2=0x33 0x15 0x06 0x03 0 0 0x10 0x08 0 0 0x11 0 0 0 0x01 0x320 0x258 0 0
Mode3=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x02 0x3C0 0x2D0 0 0
Mode4=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x02 0x400 0x300 0 0
Mode5=0x28 0x20 0x04 0xFFFC 0 0 0x20 0x0F 0 0 0x05 0 0 0 0x01 0x280 0x190 0 0x40
Mode6=0x28 0x20 0x04 0xFFFC 0 0 0x20 0x0F 0 0 0x05 0 0 0 0x01 0x280 0x1E0 0xA31B 0x40
Mode7=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x01 0x2D0 0x21C 0xA330 0x40
Mode8=0x33 0x15 0x06 0 0 0 0x10 0x08 0 0 0x12 0 0 0 0x01 0x320 0x258 0 0x40
Mode9=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x01 0x3C0 0x2D0 0 0x40
Mode10=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x01 0x400 0x300 0 0x40
Mode11=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x02 0x280 0xF0 0xA31B 0x02
Mode12=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x02 0x320 0x12C 0 0x02
Mode13=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x400 0x180 0 0x02
Mode14=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Mode15=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[ModeDef]
ModeCount=16
Mode0=0x2E 0x20 0 0 0 0 0x20 0x0F 0 0 0x12 0 0 0 0x01 0x280 0x1E0 0xA317 0
Mode1=0x8D 0x1A 0x30 0x12 0 0 0x1A 0x0D 0 0 0x12 0x14 0 0 0x01 0x2D0 0x21C 0xA330 0
Mode2=0x33 0x15 0x06 0x03 0 0 0x10 0x08 0 0 0x11 0 0 0 0x01 0x320 0x258 0 0
Mode3=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x02 0x3C0 0x2D0 0 0
Mode4=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x02 0x400 0x300 0 0
Mode5=0x28 0x20 0x04 0xFFFC 0 0 0x20 0x0F 0 0 0x05 0 0 0 0x01 0x280 0x190 0 0x40
Mode6=0x28 0x20 0x04 0xFFFC 0 0 0x20 0x0F 0 0 0x05 0 0 0 0x01 0x280 0x1E0 0xA31B 0x40
Mode7=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x01 0x2D0 0x21C 0xA330 0x40
Mode8=0x33 0x15 0x06 0 0 0 0x10 0x08 0 0 0x12 0 0 0 0x01 0x320 0x258 0 0x40
Mode9=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x01 0x3C0 0x2D0 0 0x40
Mode10=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x01 0x400 0x300 0 0x40
Mode11=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x02 0x280 0xF0 0xA31B 0x02
Mode12=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x02 0x320 0x12C 0 0x02
Mode13=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x400 0x180 0 0x02
Mode14=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Mode15=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[ColorCtrl]
Controls=0x3F 0x12 0x20 0x00 0x20 0x20 0x20

[ColorDef]
Controls=0x3F 0x12 0x20 0x00 0x20 0x20 0x20
```

第七章 PCVIDEO 源程序分析

以下我们将重点分析 GRAND VIDEO 的主要应用软件 PCVIDEO，它由 PCVIDEOC. C，FRAMEGRB. C，PCVIDEO. ASM 和 CONVERT. ASM 组成，我们只分析其中的重要的子程序：

一. PCVIDEO 初始化程序和退出程序。

二. 用一定方式俘获，显示和冻结视频，它们主要靠调用对 GRAND VIDEO 各芯片中的各寄存器赋值的底层子程序（四. 中内容）来实现。

三. 把俘获视频图像以一定格式存盘和再装入。这里牵涉到如何从物理地址在计算机内存以外的俘获缓存（扩充内存：EXTENDED MEMORY）向常规内存传送，如何进行图像变换，及存盘和它们的逆过程。

四. 通过口地址或 I2C 总线直接和寄存器打交道的，面向硬件的底层子程序。

关于图像变换的专门程序，主要是各种格式之间转换的快速算法，由于种类繁多。它们基本与硬件无关并且已有不少专书介绍，这里的 CONVERT. ASM 程序中有一部分内容就是专门进行各类图像格式变换的。我们对此作了删节，就不再进一步分析。

PCVIDEO 还考虑到 WINDOWS 应用，有关专门子程序的分析，此处从略。

以下子程序多数和 GRAND VIDEO 及 VGA 硬件有关，当读者在理解遇到困难时，可参看前几章内容。有关寄存器名称见前一章内包文件，它们的初始值见 PCVIDEO. INI。其它关于 MS C 6.0 和 MASM 5.1 的语言本身的知识读者可参看已出版的 MS C/C++ 7.0 和 MASM 6.0 的书籍。这里 MASM 还用到 CMACRO. INC 的内包文件在 MASM 6.0 的 INCLUDE 子目中。

7.1 初始化子程序，结束子程序

它是 PCVIDEO. ASM 的主要部分。PCVIDEO. ASM 的程序头是：

```
page    ,132
;Program:         PCVIDEO.DLL
;Module:          PCVIDEO.ASM
;Description:     This is the assembler source code for a Windows DLL
;                 of PC Video hardware interface routines.
;Author:          John Brothers
;Version:         1.6
;Copyright (C) Chips and Technologies, Inc. 1991, 1992
.286p
include pcvideo.inc

.xlist
```

```
include cmacros.inc      ;'C' 调用习惯宏
.list

ifdef DOSLIB
.MODEL small
.DATA
else
sBegin Data
endif

public     fInit
public     _wBoardType
public     FrameBufSel
public     NextSelIncr
public     _bHZoomFactor
public     _bVZoomFactor
public     _fFrozen
public     _fScaleVideo
fInit               dw     0          ;0=未初始化
OutWinXOrg          dw     0          ;目前显示窗 X 原点
OutWinYOrg          dw     0          ;Y 原点
OutWinXExt          dw     0          ;宽度
OutWinYExt          dw     0          ;高度
xPan                dw     0          ;显示视频在视频缓存中开始 X,Y 位置
yPan                dw     0          ;
xCapture            dw     0          ;存俘获视频到视频缓存的地址
yCapture            dw     0          ;
bHZoomFactor        db     0          ;水平放大因子.
bVZoomFactor        db     0          ;垂直放大因子.
bVZoomAdjust        db     0          ;垂直放大调节.
fScaleVideo         dw     0          ;0=最大窗 1=标定到窗口大小
fFrozen             dw     0          ;0=活动
FrameBufSel         dw     0          ;无用的 selector
NextSelIncr         dw     8          ;到下一 selector 的增量(缺省值为 8)
wBoardType          dw     0          ;1 = RGB 板 0 = YUV 板
wWriteProtectMask   dw     0FFFFh     ;写保护掩码

externW      _config
externW      iVideoMode               ;视频模式号
externW      _ModeParms               ;模式参数
externW      _wVideoWidth             ;采集视频宽度
externW      _wVideoHeight            ;高度
externW      _wVersion                ;芯片版本号
```

```
ifdef DOSLIB
externW ExtMemBfr                           ;传送到扩充内存的缓存
else
sEnd
endif

externFP PCV_SaveConfiguration              ;写配置文件到磁盘.
externFP PCV_LoadConfiguration              ;从磁盘读配置文件.

ifdef DOSLIB
.CODE
else
sBegin Code
assume cs:_text, ds:_data

externNP AllocNInitSelectors                ;分配存取视频缓存的 selector.
externNP FreeSelectors                      ;释放所有 selectors.
endif

externNP SetColor
externNP wr_i2c

ifdef DOSLIB
externNP BufferToExtended
else
externNP ComputeVideoAddress
externNP ComputeLoopParameters
endif

;****************************************
; PCV_Initialize
;用来初始化 PCVIDEO,装入配置文件,检查视频模式,对所有芯片寄存器赋值
;输入:无
;输出:有 AX=1 成功;0 失败
;            PCVIDEO 初始化
;****************************************
ExportProc PCV_Initialize
        localW   iAdjust
        localW   wCount
        localW   wColor
cBegin
        cmp      fInit, 1                   ;是否已经初始化过?
```

```
        jge     @F                      ;是
        jmp     Init                    ;不是
@@:
        add     _fInit, 1               ;初始化标帜加一.
        mov     ax,1                    ;初始化成功
        jmp     initialize_exit         ;退出
Init:
        call    PCV_LoadConfiguration   ;从文件 PCVIDEO.INI 读入配置信息
        or      ax,ax
        jge     @F                      ;如 AX<0 则
        jmp     initialize_exit         ;配置文件无效.
@@:                                     ;有效
        mov     dx,_config.wPortAddr    ;设置 PC Video 口地址
        mov     al,dl
        out     dx,al
        mov     ax,03FFh
        out     dx,ax                   ;PC Video 赋能.
        inc     dx
        in      al,dx                   ;读芯片版本号.
        shr     al,4
        xor     ah,ah
        mov     _wVersion,ax            ;除 16 后存到内存变量
get_mode_parameters:设定显示模式参数
        call    GetVideoMode            ;检测 VGA 视频模式.
        or      ax,ax                   ;要处理这显示方式吗?
        jnz     initialize_pcvideo      ;AX<>0 是的,继续.
        jmp     initialize_exit         ;AX=0 否,失败返回.
initialize_pcvideo:                     ;PCVIDEO 初始化
        lea     si,_config.PCVideoTable.aPCVideoRegs;寄存器参数表首址
        mov     cx,_config.PCVideoTable.cPCVideoRegs;寄存器参数个数
        mov     dx,_config.wPortAddr                ;寄存器地址
        rep     outsw                   ;反复送表内参数到口 OUT DX,AX
                                        ;直到 CX=0
        mov     ax,_ModeParms.ShiftClkStart
        mov     ah,SHIFT_CLK_START
        xchg    al,ah
        out     dx,ax                   ;设置移位时钟到起始位置.
        mov     dx,_config.wPortAddr
        mov     ax,_ModeParms.PaletteSkew
        shl     ax,6
        mov     ah,DW_MODE_CNTL         ;82C9001A 的显示窗模式控制寄存器
        xchg    al,ah
        out     dx,al                   ;调色板时间滞后(SKEW)
```

```
        inc     dx                      ;读回原先调色 SKEW 因子
        in      al,dx
        and     al,3fh
        or      al,ah
        out     dx,al                   ;设定调色板 skew.
        test config.wCfgFlags,CF_HASPLL;有无锁相环
        jz initialize_phixel_chips ;不需要初始锁相环.
initialize_PLL_divisor:                 ;锁相环分频初始化
        mov     dx,_config.wPortAddr
        mov     ax,PLL_DIVISOR_LO shl 8 + PLL_DIVISOR_INDEX
        out     dx,ax
        mov     bx,_ModeParms.PLLDivisor
        mov     ah,bl
        mov     al,PLL_DIVISOR_DATA
        out     dx,ax                   ;锁相环分频系数低字节
        mov     ax,PLL_DIVISOR_HI shl 8 + PLL_DIVISOR_INDEX
        out     dx,ax
        mov     ah,bh
        mov     al,PLL_DIVISOR_DATA
        out     dx,ax                   ;高字节
initialize_phixel_chips:                ;PHIXEL 芯片初始化
        mov     bx,_config.PhixelTable.cPhixelChips
        mov     wCount,bx
        lea     bx,_config.PhixelTable.aPhixelChips
initialize_phixel_chip:
        mov     cx,[bx].cPhixelRegs
        jcxz    initialize_next_phixel_chip
        lea     si,[bx].aPhixelRegs
initialize_phixel_register:             ;PHISEL 寄存器初始化
        push    cx
        push    bx
        push    si
        push    [bx].wPhixelAddr        ;PHIXEL 寄存器地址
        xor     ax,ax
        mov     al,[si].rIndex          ;指针
        push    ax
        mov     al,[si].rData           ;数据
        push    ax
        call    wr_i2c                  ;经 I2C 总线存取寄存器
        pop     si
        add     si,SIZE REGENTRY
        pop     bx
        pop     cx
```

```
        loop    initialize_phixel_register ;下一个寄存器
initialize_next_phixel_chip:               ;下一个 PHIXEL 芯片
        dec     wCount
        jz      initialize_adjust_colors
        add     bx,SIZE PHIXELCHIP
        jmp     initialize_phixel_chip
initialize_adjust_colors:                  ;颜色调节
        mov     wCount,NO_COLOR_CONTROLS      ;需调节的颜色数
        mov     wColor,BRIGHTNESS
initialize_color:
        mov     bx,wColor
        push    bx
        mov     bl,_config.bColorSettings[bx];取初始颜色设定值表中之一
        xor     bh,bh
        push    bx
        call    SetColor                   ;颜色设定
        inc     wColor
        dec     wCount
        jnz     initialize_color           ;下一种颜色
initialize_set_video_source:               ;初始设定视频源
        push    _config.wVideoSource
        call    SetVideoSource             ;选定视频源
initialize_set_video_standard:             ;初始设定视频制式
        mov     ax,_config.wCfgFlags
        and     ax,CF_NTSC                 ;NTSC 制
        push    ax
        call    SetInputFormat             ;设定输入制式
initialize_set_interlace:                  ;初始设定隔行扫描
        test    _ModeParms.fModeFlags,MF_INTERLACED;支持隔行扫描吗
        jz      initialize_compute_width
        mov     dx,_config.wPortAddr
        mov     al,INTERLACE_CNTL
        out     dx,al
        inc     dx
        in      al,dx
        or      al,INTERLACE_ON
        out     dx,al
initialize_compute_width:                  ;初始化计算宽度
        mov     dx,_config.wPortAddr
        mov     al,AW_X_START_LO           ;采集窗
        out     dx,al
        inc     dx
        in      al,dx
```

```
        mov     bl,al
        dec     dx
        mov     al,AW_X_START_HI
        out     dx,al
        inc     dx
        in      al,dx
        mov     bh,al
        and     bh,00000011b            ;BX = X 开始
        dec     dx
        mov     al,AW_X_END_HI
        out     dx,al
        inc     dx
        in      al,dx
        mov     ah,al
        and     ah,00000011b
        dec     dx
        mov     al,AW_X_END_LO
        out     dx,al
        inc     dx
        in      al,dx                   ;AX = X 结束
        sub     ax,bx
        jz      initialize_exit         ;视频宽度无效
        mov     _wVideoWidth,ax
initialize_compute_height:              ;初始化计算高度
        mov     dx,_config.wPortAddr
        mov     al,AW_Y_START_LO
        out     dx,al
        inc     dx
        in      al,dx
        mov     bl,al
        dec     dx
        mov     al,AW_Y_START_HI
        out     dx,al
        inc     dx
        in      al,dx
        mov     bh,al
        and     bh,00000011b            ;BX = Y 开始
        dec     dx
        mov     al,AW_Y_END_HI
        out     dx,al
        inc     dx
        in      al,dx
        mov     ah,al
```

```
        and         ah,00000011b
        dec         dx
        mov         al,AW_Y_END_LO
        out         dx,al
        inc         dx
        in          al,dx                ;AX = Y 结束
        sub         ax,bx
        jz          initialize_exit      ;视频高度无效
        mov         _wVideoHeight,ax
initialize_success:                      ;初始化成功
ifndef DOSLIB                            ;Windows 应用
        call        AllocNInitSelectors  ;分配扇区存视频缓存.
endif
        mov         _fInit,1             ;返回初始化成功旗
        mov         ax,1
initialize_exit:
cEnd

;*******************************************
;           PCV_Exit
;保存配置文件,退出时关闭视频.
;输出:   有
;输入:   无
;           退出 PCVIDEO
;*******************************************
ExportProc PCV_Exit
cBegin
                    cmp                  _fInit, 1
        je          @F
        sub         _fInit, 1
        jmp pcvexit
@@:
        call        PCV_SaveConfiguration ;配置信息存盘到 PCVIDEO.INI
        mov         _fInit, 0
        or          ax,ax
        jnz         @F                   ;成功
        call        TurnBorder           ;
@@:
        mov         dx,_config.wPortAddr
        mov         al,DW_MODE_CNTL      ;显示窗控制
        out         dx,al
        inc         dx
        in          al,dx
```

```
        and     al,NOT 3                ;停止显示视频窗.
        out     dx,al

ifndef DOSLIB       ;Windows 应用
        call    FreeSelectors           ;释放扇区
endif
pcvexit:

cEnd
```

初始化和结束 PCVIDEO 子程序调用如下一批子程序：

```
;*******************************************
;               GetVideoMode
;输出：     无
;输入：     无
;改变的寄存器：      AX,BX,CX,SI
;           检查 VGA 的显示模式是否和 PCVIDEO 的相容
;           参看第二章有关 TVGA CRTC 寄存器的补充知识
;*******************************************
cProc GetVideoMode,<NEAR,PUBLIC>
        localW  wHorzRes
        localW  wVertRes
cBegin

get_video_mode:
        mov     dx,3D4H                 ;VGA 的 CRT 控制器的口地址
        mov     al,1                    ;水平总计寄存器
        out     dx,al                   ;
        inc     dx
        in      al,dx                   ;读入计数值 C
        inc     al                      ;C+1
        xor     ah,ah                   ;清 AH
        shl     ax,3                    ;8*(C+1)得到水平分辨率.
        mov     wHorzRes,ax             ;保存
        mov     dx,3D4H                 ;CRTC 口地址.
        mov     al,12H                  ;求垂直分辨率.
        out     dx,al                   ;垂直显示允许结束
        inc     dx
        in      al,dx
        mov     bl,al                   ;低八位存于 BL
        mov     dx,3D4H                 ;CRTC 口地址
        mov     al,07h                  ;溢出寄存器
```

```
        out     dx,al
        inc     dx
        in      al,dx                   ;读进值在 AL 中
        xor     bh,bh                   ;BH=0
        test    al,2                    ;垂直显示允许结束的第 9 位
        jz      @F
        or      bh,1                    ;存第 9 位
@@:
        test    al,40h                  ;第 10 位
        jz      @F
        or      bh,2                    ;存 10 位
@@:
        inc     bx                      ;垂直显示允许计数 + 1
        mov     wVertRes,bx             ;得到垂直分辨率.
        mov     dx,3DAH
        in      al,dx                   ;设定属性双稳.
        mov     dx,3C0H                 ;属性控制器口地址
        mov     al,30H                  ;模式控制器
        out     dx,al
        inc     dx
        in      al,dx                   ;读模式控制寄存器.
        mov     bl,al
        xor     bh,bh
        test    bl,40h                  ;像素组方式吗?
        jz      @F                      ;不
        shr     wHorzRes,1              ;水平分辨率除以 2.
@@: lea         si,_config.aMode
        mov     cx,NO_VIDEO_MODES

check_mode:                             ;核对模式
        mov     ax,[si].VGAWidth
        cmp     ax,wHorzRes             ;核对水平分辨率
        jne     next_mode
compare_vert_res:                       ;核对垂直分辨率
        mov     ax,[si].VGAHeight
        cmp     ax,wVertRes
        jne     next_mode
check_packed_pixel:                     ;核对像素组
        mov     ax,[si].fModeFlags
        xor     ax,bx
        and     ax,MF_PACKED
        jz      have_mode
```

```
next_mode:                                  ;核对下一种模式
        add     si,SIZE MODE                ;指相参数表下一入口
        loop    check_mode

mode_not_supported:                         ;模式不行
        xor     ax,ax
        jmp     Exit_GetMode

have_mode:                                  ;选中此模式
        mov     ax,NO_VIDEO_MODES
        sub     ax,cx
        mov     _iVideoMode,ax              ;记录视频模式号.
        mov     di,offset ModeParms
        push    ds
        pop     es
        mov     cx,SIZE Mode
        rep     movsb                       ;复制模式信息.
        mov     ax, 1
Exit_GetMode:
cEnd

/*************************************************************************
 * PCV_LoadConfiguration                                                 *
 *                                                                       *
 * 装入 PCVideo 配置文件 pcvideo.ini                                      *
 *                                                                       *
 * 出口:返回码                                                            *
 *        = 1        成功                                                 *
 *        =-1        文件不存在.                                          *
 ************************************************************************/
int FAR PASCAL PCV_LoadConfiguration()
{
int fd, i, j;
int iRegCount, iChipsCount, iRegIndex=0;
WORD *Ptr;                                  /* 字指针 */
BYTE *bPtr;                                 /* 字节指针 */

#ifdef DOSLIB
        getcwd((LPSTR) &szCfgFileSpec,144);
        strcat((LPSTR) &szCfgFileSpec,"\\pcvideo.ini");/* 文件名 */
#else
        GetWindowsDirectory((LPSTR) &szCfgFileSpec,144);
        wsprintf((LPSTR) &szCfgFileSpec," %s\\pcvideo.ini",(LPSTR) &szCfgFileSpec);
```

```
#endif
    if((fd = fileopen((LPSTR) &szCfgFileSpec, F_READ)) == -1)/* 打开文件 */
     return(-1);                                   /* 文件不在 */
    fileclose(fd);
    /* Initialize PCVIDEOTBL */
    PCV_GetProfileString("pcvideo", "BufferType", szData, 10);
    if (stricmp(szData,"RGB16")==0)                /* 板型检查 */
          wBoardType=BT_RGB16;                       /* RGB16 */
    else if(stricmp(szData,"YUV411")==0)           /* YUV411 是本书介绍的类型！ */
          wBoardType=BT_YUV411;
    else return(-1);                               /* 不支持的板型 */
    PCV_GetProfileString("pcvideo", "PLL", szData, 10);
    if (stricmp(szData,"Yes")==0)                  /* 是否支持锁相环 */
          config.wCfgFlags |= CF_HASPLL;
    else config.wCfgFlags &= ~CF_HASPLL;
    PCV_GetProfileString("pcvideo", "PortAddr", szData, 10);
    config.wPortAddr=strtoul(szData, &stop,0);/* 写口地址 */
    PCV_GetProfileString("pcvideo", "VideoAddr", szData, 10);
    config.wVideoAddr=strtoul(szData, &stop,0);/* 写视频缓存地址 */
    PCV_GetProfileString("pcvideo", "VideoFormat", szData, 10);
    if ( stricmp(szData, "NTSC") == 0 )            /* 视频制式 */
          config.wCfgFlags |=1;               /* 写输入采用 NTSC 制 */
    else config.wCfgFlags |=0;            /* PAL */
    config.wVideoSource=PCV_GetProfileInt("pcvideo", "VideoSource");
                                                              /* 视频源 */
    iRegCount=config.PCVideoTable.cPCVideoRegs=
        PCV_GetProfileInt("pcvideo", "RegCount");/* 寄存器数 */
    for( i=0; i<iRegCount; i++){               /* 循环读入各寄存器的值 */
         stringf(szKey, "Reg%X", iRegIndex);
         if( PCV_GetProfileString("pcvideo", szKey, szData, 10)== 0)
            i--;                             /* 跳过不存在的寄存器 */
         else {
            config.PCVideoTable.aPCVideoRegs[i].rIndex=iRegIndex;
            config.PCVideoTable.aPCVideoRegs[i].rData=strtoul(szData,&stop,0);
            }
        iRegIndex++;                         /* 指向下一个寄存器 */
        }
    /* 初始化 PHIXEL 表 */
   iChipsCount=config.PhixelTable.cPhixelChips=
        PCV_GetProfileInt("I2CChips", "ChipsCount");
    for(i=0; i<iChipsCount; i++){
        stringf(szKey, "Chip%d", i+1);
        PCV_GetProfileString("I2CChips", szKey, szChips, 10);
```

```
        strcpy(config.PhixelTable.aPhixelChips[i].aChipID, szChips);
        PCV_GetProfileString(szChips, "I2CAddr", szData, 10);
        config.PhixelTable.aPhixelChips[i].wPhixelAddr=strtoul(szData, &stop,0);
        iRegCount=config.PhixelTable.aPhixelChips[i].cPhixelRegs=
            PCV_GetProfileInt(szChips, "RegCount");
        for( j=0; j<iRegCount;j++)   {/* Initialize PHIXELCHIP */
              stringf(szKey, "Reg%X", j);
              PCV_GetProfileString(szChips, szKey,szData,10);
              config.PhixelTable.aPhixelChips[i].aPhixelRegs[j].rIndex=j;
              config.PhixelTable.aPhixelChips[i].aPhixelRegs[j].rData=strtoul
                (szData,&stop,0);
              }
        }
/* 初始化 aMode */
for( i=0; i<NO_VIDEO_MODES; i++){
        stringf(szKey, "Mode%d", i);
        PCV_GetProfileString("ModeInfo", szKey, szBuffer,100);
        szToken=strtok(szBuffer," ");          /* 提取特征子字符串 */
        Ptr= &config.aMode[i].DispWinSkewX; /* 提取目标指针 */
        while ( szToken ! =NULL){
              *Ptr=strtoul(szToken,&stop,0);  /* 把数据变为值 */
              Ptr++;                            /* 字指针加一 */
              szToken=strtok(NULL," ");         /* 提取下一特征字符*/
              }
        }
/* 显示方式缺省表初始化 */
for( i=0; i<NO_VIDEO_MODES; i++){
        stringf(szKey, "Mode%d", i);
        PCV_GetProfileString("ModeDef", szKey, szBuffer,100);
        szToken=strtok(szBuffer," ");          /* 提取特征字符串 */
        Ptr= &aDefMode[i].DispWinSkewX;        /* 提取目标指针 */
        while ( szToken ! =NULL){
              *Ptr=strtoul(szToken,&stop,0);  /* 把数据变为数值 */
              Ptr++;                            /* 字指针加一 */
              szToken=strtok(NULL," ");         /* 提取下一特征字符串*/
              }
        }
/* 初始彩色控制 */
        PCV_GetProfileString("ColorCtrl", "Controls", szBuffer,40);
        szToken = strtok(szBuffer," ");       /*提取特征字符串 */
        bPtr= config.bColorSettings;
        while ( szToken ! =NULL){
              *bPtr=strtoul(szToken,&stop,0); /* 把数据变为数值 */
```

```
            bPtr++;                              /* 字节指针加一 */
            szToken=strtok(NULL," ");           /* 提取下一特征字符串 */
            }
    /* Initialize color defaults table */
        PCV_GetProfileString("ColorDef", "Controls", szBuffer,40);
        szToken = strtok(szBuffer," ");      /* 提取特征字符串 */
        bPtr= &aDefColorTbl[0];
        while ( szToken ! =NULL){
            *bPtr=strtoul(szToken,&stop,0); /* 把数据变为数值 */
            bPtr++;                              /* 字节指针加一 */
            szToken=strtok(NULL," ");           /* 提取下一特征字符串 */
            }
    return(1);
}

/*****************************************************************************
 * PCV_SaveConfiguration                                                      *
 *                                                                            *
 * 存系统的配置信息到 pcvideo.ini 文件.                                          *
 *                                                                            *
 * 出口:返回码                                                                 *
 *          =1      成功                                                       *
 *          =0      文件不存在.                                                 *
 *                                                                            *
 * 备注:  配置表的结构参看 PCVIDEO.H 内包文件.                                   *
 ****************************************************************************/
int FAR PASCAL PCV_SaveConfiguration()
{
int     fd, i,j;
int     iRegCount, iChipsCount;
WORD      *Ptr;                     /* Word pointer */
BYTE      *bPtr;                    /* Byte pointer */
    remove(szCfgFileSpec);
    if ((fd = filecreat((LPSTR) &szCfgFileSpec, F_MODE)) == -1)
        return 0;   /* 不能打开文件 */
    /* 存 PCVIDEO 信息 */
    stringf(szBuffer,"[PCVIDEO]\r\n");
    if( filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer)) ! =
       strlen(szBuffer))
       return 0;
    if( (wBoardType && BT_RGB16 ) == 1 )        /* RGB format */
        stringf(szBuffer, "BufferType=RGB16\r\n");
    else                                  /* 板型为 YUV 格式 */
```

```
    stringf(szBuffer, "BufferType=YUV411\r\n");
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer)) ! =
   strlen(szBuffer))
   return 0;
if( (config.wCfgFlags & CF_HASPLL) == CF_HASPLL )
    stringf(szBuffer, "PLL=Yes\r\n");
else stringf(szBuffer, "PLL=No\r\n");
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
   strlen(szBuffer))
   return 0;
stringf(szBuffer,"PortAddr=0x%X\r\n", config.wPortAddr);
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
   strlen(szBuffer))
   return 0;
stringf(szBuffer, "VideoAddr=0x%02X\r\n", config.wVideoAddr);
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
   strlen(szBuffer))
   return 0;
if( (config.wCfgFlags & 1 ) == 1 ) /* NTSC 制式 */
    stringf(szBuffer, "VideoFormat=NTSC\r\n");
else                                  /* PAL format */
    stringf(szBuffer, "VideoFormat=PAL\r\n");
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
   strlen(szBuffer))
   return 0;
stringf(szBuffer, "VideoSource=%d\r\n", config.wVideoSource);
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
   strlen(szBuffer))
   return 0;
stringf(szBuffer, "RegCount=%d\r\n", config.PCVideoTable.cPCVideoRegs);
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
   strlen(szBuffer))
   return 0;
/* 存 PCVIDEO 寄存器值 */
iRegCount=config.PCVideoTable.cPCVideoRegs;
for( i=0; i<iRegCount; i++){
   stringf(szBuffer, "Reg%X=0x%02X\r\n",
       config.PCVideoTable.aPCVideoRegs[i].rIndex,
       config.PCVideoTable.aPCVideoRegs[i].rData );
   if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
       strlen(szBuffer))
       return 0;
       }
```

```
if(filewrite(fd, "\r\n", 2)! =2)
    return 0;
/* 存 Phixel 芯片信息 */
stringf(szBuffer, "[I2CChips]\r\n");
filewrite(fd, (LPSTR)szBuffer, strlen(szBuffer));
iChipsCount=config. PhixelTable. cPhixelChips;
stringf(szBuffer, "ChipsCount=%d\r\n", iChipsCount);
filewrite(fd, (LPSTR)szBuffer, strlen(szBuffer));
for(i=0; i< iChipsCount; i++){
  stringf(szBuffer, "Chip%d=%s\r\n", i+1,(LPSTR)
  config. PhixelTable. aPhixelChips[i]. aChipID);
  if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
       strlen(szBuffer))
     return 0;
     }
if(filewrite(fd, "\r\n", 2)! =2)
    return 0;
/* 存 Phixel 芯片寄存器值 */
for(i=0; i< iChipsCount; i++){
  stringf(szBuffer, "[%s]\r\n",(LPSTR) config.
  PhixelTable. aPhixelChips[i]. aChipID);
  if(filewrite(fd, (LPSTR)szBuffer, strlen(szBuffer)) ! =
       strlen(szBuffer))
     return 0;
  stringf(szBuffer, "I2CAddr=0x%X\r\n", config.
  PhixelTable. aPhixelChips[i]. wPhixelAddr);
  if(filewrite(fd, (LPSTR)szBuffer, strlen(szBuffer)) ! =
       strlen(szBuffer))
     return 0;
  stringf(szBuffer, "RegCount=%d\r\n", config.
  PhixelTable. aPhixelChips[i]. cPhixelRegs);
  if(filewrite(fd, (LPSTR)szBuffer, strlen(szBuffer)) ! =
       strlen(szBuffer))
     return 0;
  iRegCount=config. PhixelTable. aPhixelChips[i]. cPhixelRegs;
  for( j=0; j<iRegCount; j++){           /* 存寄存器值 */
    stringf(szBuffer, "Reg%X=0x%02X\r\n",
       config. PhixelTable. aPhixelChips[i]. aPhixelRegs[j]. rIndex,
       config. PhixelTable. aPhixelChips[i]. aPhixelRegs[j]. rData );
    if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
       strlen(szBuffer))
       return 0;
       }
```

```
  if(filewrite(fd, "\r\n", 2)! =2)
     return 0;
  }
/* 存显示模式信息 */
stringf(szBuffer, "[ModeInfo]\r\n");
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
     strlen(szBuffer))
   return 0;
stringf(szBuffer, "ModeCount=%d\r\n", NO_VIDEO_MODES);
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
        strlen(szBuffer))
   return 0;
Ptr= & config.aMode[0].DispWinSkewX;              /* 提取源指针 */
for(i=0;i< NO_VIDEO_MODES;i++){                    /* 设定模式表 */
   stringf(szBuffer,"Mode%d=",i);
   for ( j=0;j<sizeof(MODE)/2;j++){                /* 格式化模式数据 */
     if ( *Ptr == (WORD) 0 )
        stringf(szData, "%d ", *Ptr);
     else
        stringf(szData, "0x%02X ", *Ptr);
     strcat(szBuffer, szData);
     Ptr++;                                        /* 指针加一 */
     }
   strcat(szBuffer, "\r\n");                       /* 模式参数结尾 */
   if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
       strlen(szBuffer))
      return 0;
   }
if(filewrite(fd, "\r\n", 2)! =2)
   return 0;
/* 存模式缺省值 */
stringf(szBuffer, "[ModeDef]\r\n");
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
     strlen(szBuffer))
   return 0;
stringf(szBuffer, "ModeCount=%d\r\n", NO_VIDEO_MODES);
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
     strlen(szBuffer))
   return 0;
Ptr= & aDefMode[0].DispWinSkewX;                  /* 提取源指针 */
for(i=0;i< NO_VIDEO_MODES;i++){                    /* 设定模式表 */
   stringf(szBuffer,"Mode%d=",i);
   for ( j=0;j<sizeof(MODE)/2;j++){               /* 格式化模式数据 */
```

```
        if ( *Ptr == (WORD) 0 )
            stringf(szData, "%d ", *Ptr);
        else
            stringf(szData, "0x%02X ", *Ptr);
        strcat(szBuffer, szData);
        Ptr++;                                      /* 指针加一 */
        }
    strcat(szBuffer, "\r\n");                       /* 模式参数结尾 */
    if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
        strlen(szBuffer))
        return 0;
    }
if(filewrite(fd, "\r\n", 2)! =2)
    return 0;
/* 存彩色信息 */
stringf(szBuffer, "[ColorCtrl]\r\n");               /* 设定彩色表 */
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
    strlen(szBuffer))
    return 0;
strcpy(szBuffer,"Controls=");
bPtr= config.bColorSettings;                       /* 提取源指针 */
for ( i=0; i<7; i++){                              /* 格式彩色控制参数 */
    stringf(szData, "0x%02X ", *bPtr);
    strcat(szBuffer, szData);
    bPtr++;
    }
strcat(szBuffer, "\r\n");                          /* 模式参数结束 */
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
    strlen(szBuffer))
    return 0;
if(filewrite(fd, "\r\n", 2)! =2)
    return 0;
/* 存彩色缺省表 */
stringf(szBuffer, "[ColorDef]\r\n");               /* 设定彩色表 */
if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
    strlen(szBuffer))
    return 0;
strcpy(szBuffer,"Controls=");
bPtr= &aDefColorTbl[0];                            /* 提取源指针 */
for ( i=0; i<7; i++){                              /* 格式彩色控制数据 */
    stringf(szData, "0x%02X ", *bPtr);
    strcat(szBuffer, szData);
    bPtr++;
```

```
            }
        if(filewrite(fd, (LPSTR) szBuffer, strlen(szBuffer))! =
            strlen(szBuffer))
          return 0;
        fileclose(fd);                                  /* 关闭 PCVIDEO.ini 文件 */
        return(1);                                      /* 成功返回 */
}

/****************************************************************************
 * PCV_GetProfileInt                                                        *
 *                                                                          *
 * 返回 pcvideo.ini 文件中整数键.                                              *
 *                                                                          *
 * 出口：返回代码                                                             *
 *         = 键的值           成功                                           *
 *         =-1                键不存在.                                      *
 *                                                                          *
 ****************************************************************************/
char    szSectionKey[20];                   /* 输入字符串缓存 */
LPSTR   szPtr;
WORD FAR PASCAL PCV_GetProfileInt(LPSTR lpSectionName, LPSTR lpKeyName)
{

     if( PCV_GetProfileString(lpSectionName, lpKeyName, szData, 10) ! = 0 )
         return(strtoul(szData,&stop,0));           /* 返回值 */
     else
         return(-1);

}

/****************************************************************************
 * PCV_GetProfileString                                                     *
 *                                                                          *
 * 读出 pcvideo.ini 文件中的字符串(原程序误作"写"! —作者).                      *
 *                                                                          *
 ****************************************************************************/
int FAR PASCAL PCV_GetProfileString(LPSTR lpSectionName,
     LPSTR lpKeyName, LPSTR lpReturnedString, int nSize)
{
FILE    *fd;
int     stringlength;

     if ((fd = fopen(szCfgFileSpec, "r")) == NULL)
```

```
        return(0);                /* 文件不在 */

    stringf(szSectionKey,"[%s]", (LPSTR) lpSectionName);
    stringlength= strlen(szSectionKey)+1;
    while( fgets(szBuffer, stringlength, fd) ! = NULL){
        if(stringcmp(szSectionKey, szBuffer) == 0){    /* 字段(Section) 匹配 */
          szPtr=lpSectionName;                        /* 初始化 ptr */
          while( stringcmp( (LPSTR) szPtr, lpKeyName) ! = 0 ){ /* 键匹配 */
              fgets(szBuffer, 100, fd);
              if( strrchr( szBuffer, '[' ) ! = NULL ){ /* 字段(Section)结尾 */
                  fclose(fd);
                  return(0);
              }
              szPtr=strtok(szBuffer,"=");             /* 提取键 */
          }
          fclose(fd);
          szPtr=strtok(NULL,"\n");                    /* 提取字串 */
          stringcpy(lpReturnedString, szPtr);
          return(stringlen(lpReturnedString));
        }
    }
    fclose(fd);
    return(0);
}
```

7.2 视频俘获,标定裁剪和显示子程序

```
;*******************************************
; 在标明位置上创建视频窗.
; 入口:        xOrg, yOrg (像素在显示屏上的绝对座标),
;              xExt, yExt (像素点)
;              fFitInWindow
;*******************************************
ExportProc PCV_CreateWindow,<SI,DI>
        parmW   xOrg                ;VGA 显示窗的左边
        parmW   yOrg                ;VGA        上边
        parmW   xExt                ;视窗宽度
        parmW   yExt                ;高度
        parmW   fFitInWindow        ;1=按视频窗标定视频

cBegin
        mov     ax,xOrg             ;保存窗口位置.
```

```
        mov     OutWinXOrg,ax
        mov     ax,yOrg
        mov     OutWinYOrg,ax
        mov     ax,xExt                 ;保存窗口大小.
        mov     OutWinXExt,ax
        mov     ax,yExt
        mov     OutWinYExt,ax
        mov     ax,fFitInWindow
        mov     _fScaleVideo,ax
        call    UpdateVideoWindow       ;更新视频窗
cEnd
;*******************************************************************************
; PCV_SetWindowPosition
;
; 移动视频窗到所标明的位置.
;
; Exported:     Yes
; 入口:         xOrg, yOrg (显示屏绝对像素点数)
;*******************************************************************************
ExportProc PCV_SetWindowPosition,<SI,DI>
        parmW   xOrg
        parmW   yOrg
cBegin
        mov     ax,xOrg                         ;保存新位置.
        mov     OutWinXOrg,ax
        mov     ax,yOrg
        mov     OutWinYOrg,ax
        call    UpdateVideoWindow
cEnd
;*******************************************************************************

; PCV_SetWindowSize
;
; 改变视频窗的宽度和高度.还可标定或不标定视频
;
; Exported:     Yes
; Entry:        xsExt, ysOrg (pixels)
;*******************************************************************************
ExportProc PCV_SetWindowSize,<FAR, PUBLIC>,<SI,DI>
        parmW   xsExt
        parmW   ysExt
        parmW   fFitInWindow
```

```
cBegin
        mov     ax,xsExt                    ;保存新的宽度和高度.
        mov     OutWinXExt,ax
        mov     ax,ysExt
        mov     OutWinYExt,ax
        mov     ax,fFitInWindow
        mov     _fScaleVideo,ax
        call    UpdateVideoWindow
cEnd
;****************************************************************************
; PCV_HorizontalZoom
;
; 设定水平放大因子.
;
; Exported:     Yes
; Entry:        iZoomFactor = 0 (1x), 1 (2x), 2(4x), or 3(8x)
;****************************************************************************
ExportProc PCV_HorizontalZoom
        parmW   iZoomFactor
cBegin
        mov     ax,iZoomFactor
        mov     ah,al
        mov     _bHZoomFactor,ah
        call    HorizontalZoom
        call    UpdateVideoWindow
cEnd
;****************************************************************************
; PCV_VerticalZoom
;
; 设定垂直放大因子.
; 输出:   有
; 输入:         iZoomFactor = 0 (1 倍), 1 (2 倍), 2(4 倍), 或 3(8 倍)
;****************************************************************************
ExportProc PCV_VerticalZoom
        parmW   iZoomFactor
cBegin
        mov     ax,iZoomFactor
        mov     ah,al
        mov     _bVZoomFactor,ah
        add     ah,bVZoomAdjust
        call    VerticalZoom
        call    UpdateVideoWindow
cEnd
```

```
;**************************************************
; PCV_EnableFieldReplication
;
; 允许帧复制消除闪烁效应.
;
; 输出:    有
; 输入:    无
;**************************************************
ExportProc PCV_EnableFieldReplication
cBegin
        or      config.wCfgFlags,CF_REPLICATE
        cmp     _fScaleVideo,0
        jne     @F
        call    EnableFieldReplicate
@@:     call    UpdateVideoWindow
cEnd
;**************************************************
'; 不许帧复制.
ExportProc PCV_DisableFieldReplication
        and     _config.wCfgFlags,NOT CF_REPLICATE
        cmp     _fScaleVideo,0
        jne     @F
        call    DisableFieldReplicate
@@:     mov     bVZoomAdjust,0          ;复位 zoom adjust.
        mov     ah,_bVZoomFactor
        call    VerticalZoom
        call    UpdateVideoWindow
cEnd
;**************************************************
; UpdateVideoWindow
;
; 内部子程序用来更新视频显示窗当 skew 因子,输出标度等改变时
;
;**************************************************
public UpdateVideoWindow
UpdateVideoWindow proc near
        cmp     fFrozen,0               ;视频要冻结吗?
        jne     uvw_continue            ;是,不需要再冻结了.
        cmp     _wVersion,1             ;能检查回扫吗?
        jl      uvw_freeze
        call    WaitVGAVerticalRetraceStart;等到 VGA 垂直回扫开始
        jmp     uvw_continue
uvw_freeze:
```

```
        call    FreezeVideo             ;停止图像采集.
uvw_continue:
        cmp     _fScaleVideo,0          ;是否标定视频?
        jz      update_unscaled_video   ;不
        test    _config.wCfgFlags,CF_REPLICATE;能否复制场
        jz      update_scaled_video     ;不.
        call    DisableFieldReplicate   ;停止复制场
        cmp     _bVZoomFactor,3         ;放大倍数是最大吗?
        je      update_scaled_video     ;不仿真复制场.
uvw_field_mode:
        mov     dx,_config.wPortAddr    ;只采集一帧.
        mov     al,AV_MODE_CNTL
        out     dx,al
        inc     dx
        in      al,dx
        or      al,4                    ;打开帧采集.
        out     dx,al
uvw_vertical_zoom:
        mov     bl,_bVZoomFactor
        xor     bh,bh                   ;调接垂直放大
        mov     ax,OutWinYExt
        shl     ax,1                    ;显示窗高度的二倍
        cmp     ax,_wVideoHeight        ;是否 Y 要减小二倍以上?
        jle     uvw_set_zoom_level      ;是，不再放大更多.
        inc     bh                      ;垂直放大因子加一.
uvw_set_zoom_level:
        mov     bVZoomAdjust,bh
        add     bl,bh                   ;垂直放大因子
        mov     ah,bl
        call    VerticalZoom            ;垂直放大
update_scaled_video:
        call    FitVideo                ;视频匹配窗口.
        jmp     uvw_wait
update_unscaled_video:
        mov     bVZoomAdjust,0
        mov     ah,_bVZoomFactor
        call    VerticalZoom            ;垂直放大
        call    UnScaleVideo            ;不标定视频.
        test    _config.wCfgFlags,CF_REPLICATE
        jz      uvw_wait
        call    EnableFieldReplicate    ;复制场
uvw_wait:
        cmp     _fFrozen,0              ;要冻结视频吗?
```

```
        jne     uvw_exit                ;是，不解冻.
        call    UnFreezeVideo           ;解冻
        cmp     _wVersion,1             ;能检查回程吗?
        jl      uvw_exit
        call    WaitToAcquireField      ;等待一帧被采集完.
uvw_exit:
        ret
UpdateVideoWindow endp
;*******************************************************************************

; PCV_SetDisplayWindow
;
; 设定显示窗寄存器.
;
; 输出:     有
; 输入:     新显示窗的 X,Y,xExt,yExt.
;*******************************************************************************
ExportProc PCV_SetDisplayWindow,<FAR,PUBLIC>,<SI,DI>
        parmW   XX
        parmW   YY
        parmW   xExt
        parmW   yExt
cBegin
        mov     ax,XX                   ;存入显示窗位置.
        mov     OutWinXOrg,ax
        mov     ax,YY
        mov     OutWinYOrg,ax
        mov     ax,xExt                 ;存入显示窗大小.
        mov     OutWinXExt,ax
        mov     ax,yExt
        mov     OutWinYExt,ax
        cmp     _wVersion,1             ;是否能检查回程?
        jl      @F
        call    WaitVGAVerticalRetraceStart;等待 VGA 垂直回扫开始
@@:     push    XX
        push    YY
        push    xExt
        push    yExt
        call    SetDisplayWindow        ;设定显示窗
cEnd
;*******************************************************************************

; PCV_SetVideoScaling
```

```
;
;设定视窗标定及选用标定或不标定.
;
;输出:    有
;输入:    fFitInWindow
;*****************************************
ExportProc PCV_SetVideoScaling,<FAR, PUBLIC>,<SI,DI>
        parmW   xSize
        parmW   ySize
        parmW   fFitInWindow

cBegin
        cmp     _wVersion,1             ;版本号=1?
        jge     sv_wait_for_retrace     ;版本号大于0,可以等待回扫.
        cmp     _fFrozen,1              ;视窗已冻结了吗?
        je      sv_continue             ;Yes, no need to freeze.
        call    FreezeVideo             ;冻结它
        jmp     sv_continue

sv_wait_for_retrace:
        call    WaitVideoVerticalRetraceStart;等待视频垂直回扫

sv_continue:
        mov     ax,fFitInWindow
        mov     _fScaleVideo,ax
        or      ax,ax
        jnz     sv_enable_scaling

sv_disable_scaling:                     ;不标定
        mov     bVZoomAdjust,0
        mov     ah,_bVZoomFactor
        call    VerticalZoom            ;垂直放大
        call    DisableScaling
        test    _config.wCfgFlags,CF_REPLICATE
        jz      sv_wait
        call    EnableFieldReplicate
        jmp     sv_wait

sv_enable_scaling:                      ;标定
        mov     bVZoomAdjust,0
        mov     ah,_bVZoomFactor
        test    _config.wCfgFlags,CF_REPLICATE
        jz      @F
```

```
        call    DisableFieldReplicate
        mov     ah,_bVZoomFactor
        cmp     ah,3                    ;放大级别是最大吗?
        je      @F                      ;不仿真复制场.
        mov     bx,OutWinYExt
        add     bx,bx
        cmp     bx,_wVideoHeight        ;高度大于 1/2 吗?
        jle     @F
        mov     bVZoomAdjust,1          ;上调节放大倍数 1.
        inc     ah

@@:     call    VerticalZoom
        push    xSize
        push    ySize
        call    ScaleVideo

sv_wait:
        cmp     _wVersion,1             ;能检查回扫吗?
        jge     sv_acquire_field
        cmp     _fFrozen,1              ;视频要冻结吗?
        je      sv_exit                 ;是,不再解冻.
        call    UnFreezeVideo           ;否,解冻.
        jmp     sv_exit

sv_acquire_field:
        call    WaitToAcquireField      ;等待采集完一场.

sv_exit:
cEnd
;*********************************************
; PCV_PanWindow
;
; 改变在显示窗中显示的视频缓存那部分.
;
; 输出:   无
; 输入:   图像左边在缓存中的 X 座标
;              底              Y
;*********************************************
ExportProc PCV_PanWindow,<SI,DI>
        parmW   X
        parmW   Y

cBegin
```

```
        cmp     _wVersion,1           ;能检查回扫吗?
        jl      @F
        call    WaitVGAVerticalRetraceStart
@@:     mov     ax,X
        mov     xPan,ax
        mov     ax,Y
        mov     yPan,ax

        mov     ax,xPan
        neg     ax
        add     ax,OutWinXOrg
        push    ax

        mov     ax,yPan
        neg     ax
        add     ax,OutWinYOrg
        push    ax

        call    SetDisplayPosition    ;复位显示位置.

pw_exit:
cEnd
;*******************************************
; PCV_SetCaptureAddress                 MASM
;
; 为俘获视频信号设置视频缓存地址.
;
; 输出:    有
; 输入:    俘获缓存中 X,Y 座标.
ExportProc PCV_SetCaptureAddress,<FAR,PUBLIC>,<SI,DI>
        parmW   XX
        parmW   YY

cBegin
        mov     ax,XX
        mov     xCapture,ax
        push    ax
        mov     ax,YY
        mov     yCapture,ax
        push    ax
        call    SetCaptureAddress
cEnd
;*******************************************
```

```
;       PCV_ClearVideoRect

;清除视频缓存所标明的矩形区.
;
;输出:   有
;输入:   rxOrg,ryOrg      矩形区左顶端
;,       rxExt,ryExt      矩形区宽度和高度
;*************************************************
ExportProc PCV_ClearVideoRect, <SI,DI,ES>
        parmW       X                          ;视频缓存中 X 位置
        parmW       Y                          ;视频缓存中 Y 位置
        parmW       xExt                       ;图像宽度
        parmW       yExt                       ;高度

        localW      cScans
        localB      fUnfreeze

ifdef DOSLIB
        localW      yPosition
else
        localW      cScansInSeg
        localW      wSegIndex
endif
cBegin
        mov         dx,_config.wPortAddr       ;视频是否已冻结.
        mov         al,AV_MODE_CNTL            ;采集模式控制寄存器
        out         dx,al
        inc         dx
        in          al,dx                      ;视频被冻结了吗?
        and         al,1
        mov         fUnfreeze,al               ;记录下来.
        jz          @F
        call        FreezeVideo                ;要存取缓存输入必需冻结.

@@:     cmp         xExt,1024
        jle         @F
        mov         xExt,1024
@@:     cmp         yExt,512
        jle         cvb_clear_rect
        mov         yExt,512

cvb_clear_rect:
ifdef DOSLIB
```

```
        xor     ax,ax
        mov     cx,EXT_MEM_BFR_SIZE/2
        mov     di,offset ExtMemBfr
        rep     stosw                   ;清扩充内存缓存.
        mov     ax,yExt
        mov     cScans,ax
        mov     ax,Y
        mov     yPosition,ax

cvb_clear_scan:
        push    X
        push    yPosition
        push    xExt
        push    1
        call    BufferToExtended        ;把数据拷贝到视频缓存.
        inc     yPosition
        dec     cScans
        jnz     cvb_clear_scan          ;清下一次扫描.
else
        push    X
        push    Y
        call    ComputeVideoAddress     ;视频缓存地址到 AX:DX
        mov     es,ax
        mov     di,dx
        mov     ax,yExt
        mov     cScans,ax
        push    xExt
        push    yExt
        call    ComputeLoopParameters   ;BX = 扫描增量, AX = 内循环计数
        mov     ax,Y
        shr     ax,5                    ;每段 32 次扫描.
        mov     wSegIndex,ax            ;开始段号.
        mov     ax,Y
        and     ax,1Fh                  ;Y 32 的余数
        sub     ax,20h
        neg     ax
        sub     cScans,ax
        jge     @F
        mov     ax,yExt
        mov     cScans,0
@@:     mov     cScansInSeg,ax
        xor     ax,ax
```

```
cvb_clear_scan:
        mov     cx,xExt                 ;每次扫描所清字数
        rep     stosw                   ;扫描清除.
        add     di,bx                   ;视频缓存中下一次扫描
        dec     cScansInSeg             ;段中扫描递减数.
        jnz     cvb_clear_scan

        cmp     cScans,0                ;所有扫描都被清除了吗?
        jle     cvb_exit                ;是，退出.

cvb_next_segment:
        mov     ax,20h                  ;更新段号.
        sub     cScans,ax
        jge     @F
        mov     ax,cScans               ;最后一段清除需小于 32 扫描线
        add     ax,20h
@@:     mov     cScansInSeg,ax
        inc     wSegIndex
        mov     ax,wSegIndex
        mul     NextSelIncr
        add     ax,FrameBufSel          ;Selector
        mov     es,ax
        xor     ax,ax
        jmp     cvb_clear_scan
endif

cvb_exit:
        test    fUnfreeze,1             ;如果曾冻结则采集.
        jz      @F
        call    UnFreezeVideo
@@:     mov     ax,1                    ;成功返回.
cEnd

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;       FitVideo

;在显示屏上把定的视频大小合适地显示在窗口上
;输出:   无
;输入:   OutWinXOrg, OutWinYOrg, OutWinXExt, OutWinYExt
;        InVideoXExt, InVideoYExt
;改变:   AX,BX,CX,DX,FLAGS
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
public  FitVideo
```

```
FitVideo proc      near
         push      OutWinXExt
         push      OutWinYExt
         call      ScaleVideo            ;设定标定寄存器.
         push      xCapture
         push      yCapture
         call      SetCaptureAddress     ;设定俘获地址.
         push      OutWinXOrg            ;设定俘获缓存器中开始显示视频数据 的地址
         push      OutWinYOrg
         call      SetDisplayPosition
         push      OutWinXOrg            ;VGA 视频显示窗的 X,Y
         push      OutWinYOrg
         push      OutWinXExt            ;宽度和高度.
         push      OutWinYExt
         call      SetDisplayWindow      ;编程显示窗口寄存器.
         ret
FitVideo endp


;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; .      UnScaleVideo

;设定 PC Video 寄存器值显示最大的视频窗,如果超过显示窗,则显示可容纳下
;的那部分.
;不允许水平和垂直标定可得到最大视频窗.视窗口在屏上位置而定显示地址和
;显示窗口控制 1 寄存器的值.俘获地址虽可设定但是一般定为零.PC Video 的寄存器
;正在设定时,视频要冻结
;输出:    无
;入口:    OutWinXOrg, OutWinYOrg, OutWinXExt, OutWinYExt
;         InVideoXExt, InVideoYExt, xPan, yPan are set.
;改变:    AX,BX,CX,DX,FLAGS
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
public  UnScaleVideo
UnScaleVideo       roc     near
         call      DisableScaling           ;关掉 X 和 Y 标定.
         push      xCapture                 ;视频缓存开始存储俘获视频数据的 X,Y 位置
         push      yCapture                 ;.
         call      SetCaptureAddress        ;设定视频缓存中俘获地址.
         push      OutWinXOrg               ;VGA 视频显示窗中 X,Y 位置
         push      OutWinYOrg               ;
         push      OutWinXExt               ;宽度和高度.
         push      OutWinYExt
         call      SetDisplayWindow         ;设定显示窗寄存器.
```

```
        mov        ax,xPan
        neg        ax
        add        ax,OutWinXOrg
        push       ax

        mov        ax,yPan
        neg        ax
        add        ax,OutWinYOrg
        push       ax
        call       SetDisplayPosition
        ret
UnScaleVideo       endp

;*************************************************************************
; FreezeVideo
; 停止视频采集.一直等到视频被冻结才返回.如果有视频回扫信号则与它同步
; 改变:    AX,DX
;*************************************************************************
public  FreezeVideo
FreezeVideo        proc     near
        cmp        _wVersion,1              ;能检测回扫吗?
        jl         @F                       ;不能,继续.
        call       WaitVideoVerticalRetraceStart
        call       WaitToAcquireField       ;等待一场被采集.
@@:       call     FreezeVideoUnsynchronized
        ret
FreezeVideo        endp

;......

ifndef DOSLIB      ;PCVIDEO.ASM 结尾
sEnd
endif
end
```

7.3 关于活动视频图像存储和重装的子程序

以 FRAMEGRB.C 为主调用 CONVERT.ASM PCVIDEO 完成活动图像的俘获,变缓和存盘。以下是 FRAMEGRB.C 程序:

```
/*******************************************************************
 * Program:PCVIDEO.DLL                                             *
```

```
* Module:FRAMEGRB.C                                                     *
* Description: C source code for saving and loading images.             *
* Author: John Brothers                                                 *
* Version: 1.6                                                          *
* Copyright (C) Chips and Technologies, Inc. 1991, 1992                 *
************************************************************************/
#ifdef DOSLIB
#include <fcntl.h>
#include <malloc.h>
#include <stdio.h>
#include <sys\types.h>
#include <sys\stat.h>
#else
#include <windows.h>
#endif
#include <io.h>
#include <string.h>
#include <pcvideo.h>

RGBQUAD GrayScale128[128] = {
  0x00,0x00,0x00,0,    0x02,0x02,0x02,0,    0x04,0x04,0x04,0,    0x06,0x06,0x06,0,
  0x08,0x08,0x08,0,    0x0A,0x0A,0x0A,0,    0x0C,0x0C,0x0C,0,    0x0E,0x0E,0x0E,0,
  0x10,0x10,0x10,0,    0x12,0x12,0x12,0,    0x14,0x14,0x14,0,    0x16,0x16,0x16,0,
  0x18,0x18,0x18,0,    0x1A,0x1A,0x1A,0,    0x1C,0x1C,0x1C,0,    0x1E,0x1E,0x1E,0,
  0x20,0x20,0x20,0,    0x22,0x22,0x22,0,    0x24,0x24,0x24,0,    0x26,0x26,0x26,0,
  0x28,0x28,0x28,0,    0x2A,0x2A,0x2A,0,    0x2C,0x2C,0x2C,0,    0x2E,0x2E,0x2E,0,
  0x30,0x30,0x30,0,    0x32,0x32,0x32,0,    0x34,0x34,0x34,0,    0x36,0x36,0x36,0,
  0x38,0x38,0x38,0,    0x3A,0x3A,0x3A,0,    0x3C,0x3C,0x3C,0,    0x3E,0x3E,0x3E,0,
  0x40,0x40,0x40,0,    0x42,0x42,0x42,0,    0x44,0x44,0x44,0,    0x46,0x46,0x46,0,
  0x48,0x48,0x48,0,    0x4A,0x4A,0x4A,0,    0x4C,0x4C,0x4C,0,    0x4E,0x4E,0x4E,0,
  0x50,0x50,0x50,0,    0x52,0x52,0x52,0,    0x54,0x54,0x54,0,    0x56,0x56,0x56,0,
  0x58,0x58,0x58,0,    0x5A,0x5A,0x5A,0,    0x5C,0x5C,0x5C,0,    0x5E,0x5E,0x5E,0,
  0x60,0x60,0x60,0,    0x62,0x62,0x62,0,    0x64,0x64,0x64,0,    0x66,0x66,0x66,0,
  0x68,0x68,0x68,0,    0x6A,0x6A,0x6A,0,    0x6C,0x6C,0x6C,0,    0x6E,0x6E,0x6E,0,
  0x70,0x70,0x70,0,    0x72,0x72,0x72,0,    0x74,0x74,0x74,0,    0x76,0x76,0x76,0,
  0x78,0x78,0x78,0,    0x7A,0x7A,0x7A,0,    0x7C,0x7C,0x7C,0,    0x7E,0x7E,0x7E,0,
  0x80,0x80,0x80,0,    0x82,0x82,0x82,0,    0x84,0x84,0x84,0,    0x86,0x86,0x86,0,
  0x88,0x88,0x88,0,    0x8A,0x8A,0x8A,0,    0x8C,0x8C,0x8C,0,    0x8E,0x8E,0x8E,0,
  0x90,0x90,0x90,0,    0x92,0x92,0x92,0,    0x94,0x94,0x94,0,    0x96,0x96,0x96,0,
  0x98,0x98,0x98,0,    0x9A,0x9A,0x9A,0,    0x9C,0x9C,0x9C,0,    0x9E,0x9E,0x9E,0,
  0xA0,0xA0,0xA0,0,    0xA2,0xA2,0xA2,0,    0xA4,0xA4,0xA4,0,    0xA6,0xA6,0xA6,0,
  0xA8,0xA8,0xA8,0,    0xAA,0xAA,0xAA,0,    0xAC,0xAC,0xAC,0,    0xAE,0xAE,0xAE,0,
  0xB0,0xB0,0xB0,0,    0xB2,0xB2,0xB2,0,    0xB4,0xB4,0xB4,0,    0xB6,0xB6,0xB6,0,
```

```
    0xB8,0xB8,0xB8,0,    0xBA,0xBA,0xBA,0,    0xBC,0xBC,0xBC,0,    0xBE,0xBE,0xBE,0,
    0xC0,0xC0,0xC0,0,    0xC2,0xC2,0xC2,0,    0xC4,0xC4,0xC4,0,    0xC6,0xC6,0xC6,0,
    0xC8,0xC8,0xC8,0,    0xCA,0xCA,0xCA,0,    0xCC,0xCC,0xCC,0,    0xCE,0xCE,0xCE,0,
    0xD0,0xD0,0xD0,0,    0xD2,0xD2,0xD2,0,    0xD4,0xD4,0xD4,0,    0xD6,0xD6,0xD6,0,
    0xD8,0xD8,0xD8,0,    0xDA,0xDA,0xDA,0,    0xDC,0xDC,0xDC,0,    0xDE,0xDE,0xDE,0,
    0xE0,0xE0,0xE0,0,    0xE2,0xE2,0xE2,0,    0xE4,0xE4,0xE4,0,    0xE6,0xE6,0xE6,0,
    0xE8,0xE8,0xE8,0,    0xEA,0xEA,0xEA,0,    0xEC,0xEC,0xEC,0,    0xEE,0xEE,0xEE,0,
    0xF0,0xF0,0xF0,0,    0xF2,0xF2,0xF2,0,    0xF4,0xF4,0xF4,0,    0xF6,0xF6,0xF6,0,
    0xF8,0xF8,0xF8,0,    0xFA,0xFA,0xFA,0,    0xFC,0xFC,0xFC,0,    0xFE,0xFE,0xFE,0};

RGBQUAD Palette[256] = {0};              /* 计算调色板用空间 */
WORD ImageBuffer[2048];                  /* 图像数据用内存 */
WORD wImageWidth,wImageHeight,wImageType;
void FAR PASCAL DebugBreak();            /* 纠错 */

int ComputePalette(int, char far *, WORD);
int CreateInversePalette(int, char far *);
int NEAR PASCAL MapToPalette(int, PSTR, char far *);
int NEAR LoadPalette(int, PSTR, int, BOOL);
int NEAR PASCAL FreezeVideo(void);
int NEAR PASCAL UnFreezeVideo(void);
int NEAR PASCAL EnableFieldReplicate(void);
int NEAR PASCAL DisableFieldReplicate(void);

char far * AllocateMemory(WORD);
void FreeMemory(char far *, HANDLE);

extern CFGSTRUCT config;                 /* 配置数据结构 */
extern WORD buf[64];                     /* 数据段中所需缓存 */
extern WORD fFrozen;                     /* = 0 活动视频 */
extern WORD fScaleVideo;
BITMAPFILEHEADER DIBFileHdr={0x4D42,0,0,0,0};
BITMAPINFOHEADER BMInfoHdr={sizeof(BITMAPINFOHEADER),
    0,0,1,0,BI_RGB,0,0,0,0,0};
TRGHDRtargaHeader = {0,0,2,0,0,0,0,0,0,0,0,0,16,0x20};
HANDLEhMemory;                           /* Windows 分配的内存把柄 */


/*****************************************

程序名      PCV_SaveImageRect              C 语言
造句法   int PCV_SaveImageRect(pFile,xLift,yTop,width,heigh,
            wFileType,wOptions)
```

功能	从视频缓存中读一矩形像素块按标明的图像文件格式变换并写盘	
输入参数		类型/描述
	pFile	char * 指向文件名字符串的指针
	xLeft	int 块左边的像素座标
	yTop	int 块顶端的像素座标
	width	int 以像素点数为单位的块宽度
	height	int 以像素点数为单位的块高度
	wFileType	WORD 文件类型
	文件类型	描述
	BM_YUV411	IBM MMotion (4:1:1 YUV) 格式
	BM_DIB24	Windows 3.0 24 位设备无关位图格式(DIB)
	BM_DIB8G	Windows 3.0 DIB 8 位灰度指标
	BM_DIB8P	Windows 3.0 DIB 8 位彩色(经调色的)
	BM_TRG24	Targa 24 位格式
	BM_TRG16	Targa 16 位格式
	BM_TRG16	Targa 16 位格式
	wOptions	WORD 杂项选用参数(必需为零)
返回值	>0	原来图像中的色彩数
	0	读视频缓存出错
	=-1	建文件出错,可能磁盘出错
	=-2	内存不能分配
	=-3	写文件出错,可能磁盘出错
备注	应保证 VideoPlus 的缓存不和系统内存地址矛盾.	
它调用子程序:	FreezeVideo	;冻结视频
	UnFreezeVideo	;采集视频
	AllocateMemory	;分配内存
	FreeMemory	;释放内存
	PCV_ReadImageRect	;读图像矩形块
	其它 DOS 中断调用	;文件操作

```
*******************************************/
int FAR PASCAL PCV_SaveImageRect(LPSTR lpFile, WORD xPos, WORD yPos,
WORD width, WORD height, WORD wFileType, WORD wOptions)
{
BOOL        fInvertImage, fUnFreeze;
WORD        wPaletteEntries, i, j, scan_width;
int         yIncrement, vZoom;
int         iReturnCode, iClrsFound, fd, fdRaw, fdOut, fdPal;
HANDLE      hWorkMem;
char far * fpWorkSpace;

  width = min(width,CAPTURE_BUFFER_WIDTH);
  height = min(height,CAPTURE_BUFFER_HEIGHT);        /* 剪裁掉多余像素 */
  if (width%4 ! = 0)                                  /* 修正宽度为 4 的倍数. */
```

```
        width = width+(4-width%4);
  if((fdOut = fd = filecreat(lpFile, F_MODE)) == -1);      /* 建文件 */
        return(-1);                                         /* 出错 */
  if (! fFrozen) {                               /* 如果没有冻结视频则冻结它. */
     FreezeVideo();
     fUnFreeze = 1;
     }
  else fUnFreeze = 0;
  switch (wFileType)                             /* 不同图像文件格式分别处理 */
        {
        case BM_YUV411:                          /* IBM MMotion 格式 */
          fInvertImage = FALSE;
                                                 /* 不需反序 */
filewrite(fd, (LPSTR)MMP_HDR_STRING, strlen(MMP_HDR_STRING));/* 写文件头 */
          buf[0] = buf[1] = buf[2] = 0;
          buf[3] = width;
          buf[4] = height;     前面字节为 0,后面字节分别为图像宽度和高度
          filewrite(fd, (LPSTR)buf, 10); /* 写 10 字节 */
          scan_width = width + width/2;
          break;
        case BM_DIB24:                           /* 24 bit/pixel Windows DIB */
          fInvertImage = TRUE;                   /* Flip the image in the Y direction. */
          scan_width = 3 * (DWORD) width;        /* Bytes per scan line */
          BMInfoHdr.biBitCount = 24;
          BMInfoHdr.biClrUsed = BMInfoHdr.biClrImportant = 0; /* True color */
          DIBFileHdr.bfOffBits = sizeof(BITMAPFILEHEADER)+sizeof(BITMAPINFOHEADER);
          goto write_DIB_header;
          break;
        case BM_DIB8P:                           /* 经调色的 Windows DIB */
          fInvertImage = TRUE;                   /* 图像在 Y 方向抖动. */
          scan_width = 2 * width;                /* 每扫描一行的字节数 */
          BMInfoHdr.biBitCount = 8;
          /* 分配 64 kB 段作为工作空间. */
          if (! (fpWorkSpace = AllocateMemory(0x8000)))
            {
            iReturnCode = -2;                    /* 内存不能分配 */
            goto save_return;
            }
          else hWorkMem = hMemory;               /* 内存分配成功,hMemory 是 Windows 或 Dos
          分配内存的把柄. */
          /* 建立一未加工数据文件来保存未经调色的红绿蓝 15 位值. */
          if ((fdOut = fdRaw = filecreat("PCVIDEO.RAW", F_MODE)) == -1)
            return(-1);                          /* 建立文件出错 */
```

```
    break;
  case BM_DIB8G:                          /* Gray—scale Windows DIB */
    fInvertImage = TRUE;                  /* Flip the image in the Y direction. */
    scan_width = width;                   /* Bytes per scan line */
    BMInfoHdr.biBitCount = 8;
    BMInfoHdr.biClrUsed = 128;            /* 128 gray scale palette */
    DIBFileHdr.bfOffBits = sizeof(BITMAPFILEHEADER)+sizeof
      (BITMAPINFOHEADER)+128 * sizeof(RGBQUAD);
    write_DIB_header:
    BMInfoHdr.biSizeImage = scan_width * (DWORD) height;
    DIBFileHdr.bfSize = DIBFileHdr.bfOffBits + BMInfoHdr.biSizeImage;
    filewrite(fd,(LPSTR) &DIBFileHdr, sizeof(BITMAPFILEHEADER));
    BMInfoHdr.biSize = sizeof(BITMAPINFOHEADER);
    BMInfoHdr.biWidth = width;
    BMInfoHdr.biHeight = height;
    BMInfoHdr.biXPelsPerMeter = BMInfoHdr.biYPelsPerMeter = 0;
    filewrite(fd, (LPSTR) &BMInfoHdr, sizeof(BITMAPINFOHEADER));
    if (wFileType == BM_DIB8G)
      filewrite(fd, (LPSTR) &GrayScale128, 128 * sizeof(RGBQUAD));
    break;
  case BM_TRG32:
    fInvertImage = FALSE;                 /* Flip the image in the Y direction. */
    scan_width = 4 * width;               /* Bytes per scan line */
    targaHeader.bBitsPixel = 32;
    goto write_targa_header;
    break;
  case BM_TRG24:
    fInvertImage = FALSE;                 /* Flip the image in the Y direction. */
    scan_width = 3 * width;               /* Bytes per scan line */
    targaHeader.bBitsPixel = 24;
    goto write_targa_header;
    break;
  case BM_TRG16:
    fInvertImage = FALSE;
    scan_width = 2 * width;               /* Bytes per scan line */
    targaHeader.bBitsPixel = 16;
    write_targa_header:
    targaHeader.wWidth = width;
    targaHeader.wHeight = height;
    filewrite(fd,(LPSTR) &targaHeader, sizeof(TRGHDR));
    break;
  default:
    break;
```

```
      }                                         /* 不同格式分别处理结束 */
  yIncrement = 1;                               /* 计算一或二行,放大一或二倍 */
  vZoom = 1;
  if (config.wCfgFlags & CF_REPLICATE) {        /* 循环 */
    if (fScaleVideo)                            /* 反序:从下往上存 */
      vZoom = 2;
    else yIncrement = 2;
    }
  for (i=0; i<(height/vZoom); i+=yIncrement) {
    if (fInvertImage)
      iReturnCode = PCV_ReadImageRect((PSTR) &ImageBuffer,xPos,
          height-(yPos+i+1),width,1,
          wFileType,fpWorkSpace);               /* 正序,从上往下存 */
    else iReturnCode = PCV_ReadImageRect((PSTR)
          &ImageBuffer,xPos,yPos+i,width,1,
          wFileType,fpWorkSpace);
    if (iReturnCode ! = 1)                      /* 从视缓存读一行到暂存 */
      goto save_return;
    for (j=0; j<(yIncrement * vZoom); j++) {
      if ((unsigned short) filewrite(fdOut,(PSTR) &ImageBuffer,
        scan_width) ! = scan_width)
                                                /* 写到磁盘文件中,暂存 */
        {
        iReturnCode = -3                        /* 写文件出错 */
        goto free_memory_and_return;
        }
      }                                         /* j 循环结束 */
    }                                           /* i 循环结束 */
  /* 当场复制和标定同时赋能时,图像有奇数扫描行,应小心最后一行.   */
  if ((config.wCfgFlags & CF_REPLICATE)
&& (vZoom == 2) && (height%2 == 1)) {
          if (fInvertImage)
      iReturnCode = PCV_ReadImageRect((PSTR) &ImageBuffer,xPos,
          height-(yPos+i+1),width,1,wFileType,fpWorkSpace);
    else iReturnCode = PCV_ReadImageRect((PSTR) &ImageBuffer,xPos,yPos+i,
          width,1,wFileType,fpWorkSpace);
    if (iReturnCode ! = 1)
      goto save_return;
    if ((unsigned short) filewrite(fdOut,(PSTR)
          &ImageBuffer, scan_width) ! = scan_width)
      {
      iReturnCode = -3;
      goto free_memory_and_return;
```

```
        }
      }
   if (wFileType == BM_DIB8P) {
     fileclose(fdRaw);                          /* 关闭未加工数据文件. */
     iClrsFound = ComputePalette(MAX_PALETTE_ENTRIES,
         fpWorkSpace, wOptions);
     BMInfoHdr.biClrUsed = min(iClrsFound,MAX_PALETTE_ENTRIES);
     BMInfoHdr.biSizeImage = (DWORD) width * (DWORD) height;
     DIBFileHdr.bfOffBits = sizeof(BITMAPFILEHEADER)+sizeof(BITMAPINFOHEADER)+
                 BMInfoHdr.biClrUsed * sizeof(RGBQUAD);
     DIBFileHdr.bfSize = DIBFileHdr.bfOffBits + BMInfoHdr.biSizeImage;
     filewrite(fd,(LPSTR) &DIBFileHdr, sizeof(BITMAPFILEHEADER));
     BMInfoHdr.biWidth = width;
     BMInfoHdr.biHeight = height;
     filewrite(fd, (LPSTR) &BMInfoHdr, sizeof(BITMAPINFOHEADER));
     CreateInversePalette((WORD) BMInfoHdr.biClrUsed, fpWorkSpace);
     filewrite(fd, (LPSTR) &Palette, (WORD) BMInfoHdr.
          biClrUsed * sizeof(RGBQUAD));
     /* 如果请求输出一调色文件. */
     if (wOptions & PF_PALFILE) {
       if ((fdPal = filecreat("PCVIDEO.PAL", F_MODE)) == -1)
         return (-1);
       else {
         filewrite(fdPal,(LPSTR) "PAL",3);
         filewrite(fdPal,(LPSTR) &BMInfoHdr.biClrUsed, 2);
         filewrite(fdPal,(LPSTR) &Palette, (WORD) BMInfoHdr.
              biClrUsed * sizeof(RGBQUAD));
         fileclose(fdPal);
         }
       }
     fdRaw = fileopen("PCVIDEO.RAW", F_READ);            /* 打开文件读原始数据. */
     for (i=0; i<height; i++) {
       fileread(fdRaw,(PSTR) &ImageBuffer,2 * width);    /* 读一扫描行原始数据. */
       MapToPalette(width,(PSTR) &ImageBuffer,fpWorkSpace);
       filewrite(fd,(PSTR) &ImageBuffer,width);      /* 写经过调色的值. */
       }
     fileclose(fd);                              /* 关闭 DIB 文件. */
     fileclose(fdRaw);                           /* 关闭原始数据文件. */
     remove("PCVIDEO.RAW");                      /* 删除原始数据文件. */
     iReturnCode = iClrsFound;                   /* 返回找到的彩色数. */
     }
   free_memory_and_return:
   if (wFileType == BM_DIB8P)
```

```
    FreeMemory(fpWorkSpace,hWorkMem);                /* 释放工作内存空间. */
  save_return:
  fileclose(fd);                                     /*关闭文件*/
  if (fUnFreeze)                                     /* 如果本来没有冻结则解冻视频. */
    UnFreezeVideo();
  return(iReturnCode);
}

/*******************************************************************
程序名        PCV_LoadImageRect                     C 语言
造句法   int  PCV_LoadImageRect(pFile,xLeft,yTop)
功能          读所述图像文件，检查它的格式,变图像数据为视频缓存格式并写到
              视频缓存的 xLeft，yTop 位置.
输入参数                     类型/描述
              pFile          char * 文件名字符串的指针
              xLeft          int 块左边像素座标
              yTop           int 块顶端像素座标
返回值        1              成功
              0              写视频缓存出错
              -1             打开文件出错
              -2             不能分配内存
              -3             数据不足
              -4             不认识的的文件格式
备注          应保证 VideoPlus 的缓存不和系统内存地址矛盾
可参看        PCV_SaveImageRect,PCV_SetVideoAddress
调用子程序:   PCV_OemInit,PCV_WriteImageRect
*******************************************************************/
int FAR PASCAL PCV_LoadImageRect(LPSTR lpFile, WORD xPos, WORD yPos)
{
WORD        i, scan_width;
int         iReturnCode, fd, bitcount, clrused;
BOOL        fWindowsDIB, fInvertImage;
LPMMPHDR lpMmpHdr;
LPTRGHDR lpTrgHdr;
LPBITMAPINFOHEADER lpBMInfoHdr;
LPBITMAPCOREHEADER lpBMCoreHdr;
LPBITMAPFILEHEADER lpBMFileHdr;

  if ((fd = fileopen(lpFile, F_READ)) == -1) /* 打开文件读出错 */
        return(-1);
  /*  决定文件类型和读进相应的文件头信息 */
  if (fileread(fd,(LPSTR) &buf,sizeof
        (BITMAPFILEHEADER)+sizeof(BITMAPINFOHEADER)) ! =
```

```
    sizeof(BITMAPFILEHEADER)+sizeof(BITMAPINFOHEADER))
  {   /* 读足够的字节来决定文件类型 */
  iReturnCode = -3;               /* 数据不足 */
  goto load_return;
  }
if (strnicmp((char *) buf, MMP_HDR_STRING,
    strlen(MMP_HDR_STRING)) == 0)
  {                                   /* IBM MMotion 位图 */
  wImageType = BM_YUV411;
  fInvertImage = FALSE;
  lpMmpHdr = (LPMMPHDR) &buf;
  wImageWidth = lpMmpHdr -> mhWidth;
  wImageHeight = lpMmpHdr -> mhHeight;
  scan_width = 3 * wImageWidth/2;     /* Bytes per scan line */
  lseek(fd,sizeof(MMPHDR),SEEK_SET);
                                      /* Set file pointer to start of image data. */
  }
else if (strnicmp((char *) buf, BMP_HDR_STRING,
    strlen(BMP_HDR_STRING)) == 0)
  {   /* Windows device-independent bitmap */
  fInvertImage = TRUE;                /* Invert the image as it is read in. */
  lpBMFileHdr = (LPBITMAPFILEHEADER) &buf;
  lpBMInfoHdr = (LPBITMAPINFOHEADER) ((ADDRESS) lpBMFileHdr +
    sizeof(BITMAPFILEHEADER));
  lpBMCoreHdr = (LPBITMAPCOREHEADER) lpBMInfoHdr;
                                      /* For Presentation Manager DIBs */
  if (lpBMInfoHdr -> biSize == sizeof(BITMAPINFOHEADER)) {
    wImageWidth = lpBMInfoHdr -> biWidth;
    wImageHeight = lpBMInfoHdr -> biHeight;
    bitcount = lpBMInfoHdr -> biBitCount;
    clrused = (int) lpBMInfoHdr -> biClrUsed;
    fWindowsDIB = TRUE;           /* Windows 3.0 DIB */
    }
  else {
    wImageWidth = lpBMCoreHdr -> bcWidth;
    wImageHeight = lpBMCoreHdr -> bcHeight;
    bitcount = lpBMCoreHdr -> bcBitCount;
    clrused = 0;
    fWindowsDIB = FALSE;          /* Presentation Manager 1.2 DIB */
    lseek(fd,sizeof(BITMAPFILEHEADER)+sizeof(BITMAPCOREHEADER),SEEK_SET);
    }
  if (bitcount == 24) {
    wImageType = BM_DIB24;
```

```
        scan_width = 3 * wImageWidth;                    /* Bytes per scan line */
        }
    else if (bitcount == 8) {
        wImageType = BM_DIB8P;
        scan_width = wImageWidth;                        /* 每扫描行的字节数 */
        if (clrused == 0)
            clrused = 256;
        }
    else if (bitcount == 4) {
        wImageType = BM_DIB4D;                           /* For now treat gray scale as color */
        scan_width = wImageWidth/2;                      /* Bytes per scan line */
        if (clrused == 0)
            clrused = 16;
        }
    else {
        iReturnCode = -4;                                /* Unsupported DIB file type */
        goto load_return;
        }
    }
else {
        lpTrgHdr = (LPTRGHDR) &buf;
        wImageWidth = lpTrgHdr -> wWidth;
        wImageHeight = lpTrgHdr -> wHeight;
        if (lpTrgHdr -> bImageDescriptor & ORIGIN_TOP_LEFT)
            fInvertImage = FALSE;
        else fInvertImage = TRUE;
        if ((lpTrgHdr -> bImageType == 2) || (lpTrgHdr -> bImageType == 4)) {
            if (lpTrgHdr -> bBitsPixel == 16) {
                scan_width = 2 * wImageWidth;
                wImageType = BM_TRG16;
                }
            else if (lpTrgHdr -> bBitsPixel == 24) {
                scan_width = 3 * wImageWidth;
                wImageType = BM_TRG24;
                }
            else if (lpTrgHdr -> bBitsPixel == 32) {
                scan_width = 4 * wImageWidth;
                wImageType = BM_TRG32;
                }
            else {
                iReturnCode = -4;                        /* 不认识的数据格式 */
                goto load_return;
                }
```

```
            lseek(fd,sizeof(TRGHDR),SEEK_SET);          /* 设定文件指针到图像数据开头. */
            }
        else {
            iReturnCode = -4;                           /* 不认识的文件格式 */
            goto load_return;
            }
    }
    if ((wImageWidth > CAPTURE_BUFFER_WIDTH) ||
            (wImageHeight > CAPTURE_BUFFER_HEIGHT))
      {
      iReturnCode = 0;                                  /* 图像太大不能全放入视频缓存 */
      goto load_return;
      }
    if ((wImageType == BM_DIB8P) || (wImageType == BM_DIB4D)) {
      /* 装入调色板 */
      if (iReturnCode = LoadPalette(fd,(PSTR) &Palette, clrused,
        fWindowsDIB) ! = 1)
        goto load_return;                               /* 数据不足 */
      }
    if (config. wCfgFlags & CF_REPLICATE)
      PCV_DisableFieldReplication();
    for (i = 0; i < wImageHeight; i++) {
      if (fileread(fd,(PSTR) &ImageBuffer, scan_width) ! = scan_width)
        {
        iReturnCode = -3;                               /* 数据不足 */
        goto load_return;
        }
      if (fInvertImage)
        iReturnCode = PCV_WriteImageRect((PSTR) &ImageBuffer,(PSTR)
          &Palette,xPos,wImageHeight-
                            (yPos+i+1),wImageWidth,1,wImageType);/* 写一行到 */
      else iReturnCode = PCV_WriteImageRect((PSTR) &ImageBuffer,(PSTR)
          &Palette,xPos,yPos+i,wImageWidth,1,wImageType);
      if (iReturnCode ! = 1)
        goto load_return;
      }
    load_return:
    fileclose(fd);
    return(iReturnCode);
}
```

PCV_SaveImageRect 和 PCV_LoadImageRect 两 C 语言子程序分别调用了 PCV_ReadImageRect 和 PCV_WriteImageRect 两汇编语言子程序. 后者是 CONVERT. ASM 的一部分.

以下 CONVERT.ASM 中有关图像格式变换的内容只保留一例,其余均已略去.

```
page    ,132
; 程序:     PCVIDEO.DLL
; 模块:     CONVERT.ASM
; 描述: PC Video Windows 动态联结库.本模块包含图像变换子程序.
; Author:       John Brothers
; Version:      1.6
; Date:         April 7, 1991
; Copyright (C) Chips and Technologies, Inc. 1991, 1992

.286p
include pcvideo.inc

.xlist
include cmacros.inc                         ;'C' 调用习惯宏
.list

ifdef DOSLIB
.MODEL small
.DATA
else
sBegin  Data
endif
externW wBoardType                          ;1 = 视频缓存是 16 位 RGB.
externW Palette
externW config
externW NextSelIncr
externW FrameBufSel

ifdef DOSLIB
public DummyGDT, ExtMemBfr
DefaultGDT          db 16 dup (0)
                    db 5 dup (0), 93h, 0, 0
                    db 5 dup (0), 93h, 0, 0
                    db 16 dup (0)
DummyGDT            db size GDT dup (0)
ExtMemBfr           db EXT_MEM_BFR_SIZE dup (0)         ;扩充内存传送用
endif

; 子程序表:从视频缓存读入,变换成所需的文件格式,然后写到输出缓存

VideoToSystemTbl  dw YUV411UnPackedToRGB24          ;Windows 24 bit true color
```

```
        dw YUV411UnPackedToRGB8Color           ;Windows 8 位调色,以下仅分析此例
        dw YUV411UnPackedToRGB8Gray            ;Windows 8 bit gray—scale
        dw YUV411UnPackedToRGB4Dither          ;Windows 4 bit dithered
        dw YUV411UnPackedToRGB32               ;Targa 32 bit true color
        dw YUV411UnPackedToRGB24               ;Targa 24 bit true color
        dw YUV411UnPackedToRGB16               ;Targa 16 bit true color
        dw YUV411UnPackedToYUV411Packed        ;IBM MMotion
        dw UnimplementedFormat                 ;TIFF 8 bit color
        dw UnimplementedFormat                 ;TIFF 8 bit gray—scale
        dw UnimplementedFormat                 ;TIFF 4 bit color
        dw UnimplementedFormat                 ;PCX 8 bit color
        dw UnimplementedFormat                 ;PCX 8 bit gray—scale
        dw UnimplementedFormat                 ;PCX 4 bit color
        dw UnimplementedFormat                 ;GIF 8 bit color
        dw UnimplementedFormat                 ;GIF 8 bit gray—scale
        dw RGB16ToRGB24                        ;Windows 24 bit true color
        dw RGB16ToRGB8Color                    ;Windows 8 bit palettized
        dw RGB16ToRGB8Gray                     ;Windows 8 bit gray—scale
        dw RGB16ToRGB4Dither                   ;Windows 4 bit dithered
        dw RGB16ToRGB32                        ;Targa 32 bit true color
        dw RGB16ToRGB24                        ;Targa 24 bit true color
        dw RGB16ToRGB16                        ;Targa 16 bit true color
        dw RGB16ToYUV411Packed                 ;IBM MMotion
        dw UnimplementedFormat                 ;TIFF 8 bit color
        dw UnimplementedFormat                 ;TIFF 8 bit gray—scale
        dw UnimplementedFormat                 ;TIFF 4 bit color
        dw UnimplementedFormat                 ;PCX 8 bit color
        dw UnimplementedFormat                 ;PCX 8 bit gray—scale
        dw UnimplementedFormat                 ;PCX 4 bit color
        dw UnimplementedFormat                 ;GIF 8 bit color
        dw UnimplementedFormat                 ;GIF 8 bit gray—scale

; 子程序表:从系统内存缓存读入数据,由所述文件格式变换格式,再写到视频缓存.

SystemToVideoTbl dw RGB24ToYUV411UnPacked      ;Windows 24 bit true color
        dw RGB8ToYUV411UnPacked                ;Windows 8 bit palettized
        dw RGB8ToYUV411UnPacked                ;Windows 8 bit gray—scale
        dw RGB4ToYUV411UnPacked                ;Windows 4 bit dithered
        dw RGB32ToYUV411UnPacked               ;Targa 32 bit true color
        dw RGB24ToYUV411UnPacked               ;Targa 24 bit true color
        dw RGB16ToYUV411UnPacked               ;Targa 16 bit true color
        dw YUV411PackedToYUV411UnPacked        ;IBM MMotion
        dw UnimplementedFormat                 ;TIFF 8 bit color
```

```
                dw UnimplementedFormat          ;TIFF 8 bit gray-scale
                dw UnimplementedFormat          ;TIFF 4 bit color
                dw UnimplementedFormat          ;PCX 8 bit color
                dw UnimplementedFormat          ;PCX 8 bit gray-scale
                dw UnimplementedFormat          ;PCX 4 bit color
                dw UnimplementedFormat          ;GIF 8 bit color
                dw UnimplementedFormat          ;GIF 8 bit gray-scale
                dw RGB24ToRGB16                 ;Windows 24 bit true color
                dw RGB8ToRGB16                  ;Windows 8 bit w/palette
                dw RGB8ToRGB16                  ;Windows 8 bit w/palette
                dw RGB4ToRGB16                  ;Windows 4 bit w/palette
                dw RGB32ToRGB16                 ;Targa 32 bit true color
                dw RGB24ToRGB16                 ;Targa 24 bit true color
                dw RGB16ToRGB16                 ;Targa 16 bit true color
                dw YUV411PackedToRGB16          ;IBM MMotion
                dw UnimplementedFormat          ;TIFF 8 bit color
                dw UnimplementedFormat          ;TIFF 8 bit gray-scale
                dw UnimplementedFormat          ;TIFF 4 bit color
                dw UnimplementedFormat          ;PCX 8 bit color
                dw UnimplementedFormat          ;PCX 8 bit gray-scale
                dw UnimplementedFormat          ;PCX 4 bit color
                dw UnimplementedFormat          ;GIF 8 bit color
                dw UnimplementedFormat          ;GIF 8 bit gray-scale
ifndef DOSLIB
sEnd.
endif
externNP TurnBorder                             ;改变调色板边界色的子程序.

ifdef DOSLIB
.CODE
else
sBegin   Code
assume cs:_text, ds:_data
endif


;***************************************
;程序名          PCV_ReadImageRect                       汇编语言
;造句法      int PCV_ReadImageRect(fpBuffer,X,Y,XExt,YExt,wBitmapType,
;                fpWorkSpace)
;功能            读图像缓存矩形区数据(5:6:5 RGB 或 4:1:1 YUV)并变成指定
;                格式写到输出远指针
;输入参数                               类型/描述
```

```
;             fpBuffer                D  输出缓存的远指针
;             X                       W  图像块的左边座标
;             Y                       W  图像块的顶端座标
;             XExt                    W  块宽度
;             YExt                    W  块高度
;             wBitmapType             W  位图类型
;             fpWorkSpace             D  调色板工作区远指针
;返回值       AX=1 成功
;调用子程序:  ExtendedToBuffer,ComputeVideoAddress
;             ComputeLoopParameters
;
; YUV 格式执行方法
; 在 YUV 板视频缓存中数据按 4:1:1 YUV 格式存储.
; 每一像素有一 7 位无符号灰度值,Y.两个色差分量,U (B-Y) 和 V (R-Y),是
; 带符号的 7 位值.因为色度取样速率仅为灰度的四分之一,每四个像素只有一个色
; 度值.这两个色度在四个像素中分布如下:
;
;    字 0     15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
;             U6  U5  V6  V5  XX  XX   XX  XX  Y6  Y5  Y4  Y3  Y2  Y1  Y0  XX

;    字 1     15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
;             U4  U3  V4  V3  XX  XX   XX  XX  Y6  Y5  Y4  Y3  Y2  Y1  Y0  XX

;    字 2     15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
;             U2  U1  V2  V1  XX  XX   XX  XX  Y6  Y5  Y4  Y3  Y2  Y1  Y0  XX

;    字 3     15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
;             U0  XX  V0  XX  XX  XX   XX  XX  Y6  Y5  Y4  Y3  Y2  Y1  Y0  XX

; U 和 V 还分别按 127/179 和 127/226 标定来适配 8 位,最低位被截去以适应 7 位.
;
;       Y = 0.299*R + 0.587*G + 0.114*B
;       V = R-Y = R - 0.299*R - 0.587*G - 0.114*B
;       V = 0.701*R - 0.587*G - 0.114*B
;       Vmax = 0.701*255 - 0.587*0 - 0.114*0 = 178.76
;       Vmin = 0.701*0 - 0.587*255 - 0.114*255 = -178.76
;       因此 V 按 127/179 来标定以适应于 8 位.
;
;       U = B-Y = -0.299*R - 0.587*G + B - 0.114*B
;       U = -0.299*R - 0.587*G + 0.866:B
;       Umax = -0.299*0 - 0.587*0 + 0.866*255 = 225.93
;       Umin = -0.299*255 - 0.587*255 + 0.866*0 = -225.93
;       因此 U 按 127/226 来标定以适应于 8 位.
```

```
;
;红:从视频缓存标定值 V 乘以 179/127 后加 Y
;蓝:                  U       226/127      Y
;绿:  0.587 * G = Y - 0.299 * R - 0.114 * B
;       G = 1.706 * Y - 0.509 * R - 0.194 * B
;
;RGB 执行方法
;对 RGB 板每一像素占有 16 位,红蓝各占 5 位,绿占 6 位

;    像素    15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
;    格式    G0  R4  R3  R2  R1  R0   G5  G4  G3  G2  G1  B4  B3  B2  B1  B0

ExportProc PCV_ReadImageRect,<BX,CX,DX,SI,DI,DS,ES>
        parmW     pBuffer                    ;输出缓存的偏移地址.
        parmW     X                          ;待变换的像素块左端
        parmW     Y                          ;上端
        parmW     xExt                       ;块宽度
        parmW     yExt                       ;高度
        parmW     wBitmapType                ;位图类型.
        parmD     fpWorkSpace                ;调色用的工作空间指针

        localV    luma,4
        localW    U
        localW    V
        localW    wCount
        localV    adRGBQuad,16
        localW    npfncVideoToSystem
        localD    lpSource

cBegin
        cld
        mov       bx,wBitmapType             ;计算表指针.
        and       bx,0Fh
        mov       ax,_wBoardType
        shl       ax,4
        or        bx,ax
        shl       bx,1
        mov       ax,VideoToSystemTbl[bx]    ;变换子程序地址
        mov       npfncVideoToSystem,ax
ifdef DOSLIB
        push      X
        push      Y
        push      xExt
```

```
        push    1
        call    ExtendedToBuffer            ;从视频到缓存.
        or      ax,ax                       ;如不成功退出.
        jz      Exit_ReadImageRect
        mov     ax,ds
        mov     es,ax
        mov     di,pBuffer                  ;写数据的地方.
        mov     si,offset ExtMemBfr         ;读数据的地方.
        mov     ax,xExt
        shr     ax,2
        cmp     wBitmapType,BM_YUV411       ;如果输出文件是 4:1:1 YUV,
        je      @F                          ;一次做四像素
        cmp     _wBoardType,BT_YUV411       ;如果读 4:1:1 YUV 数据,一次读四像素
        je      @F
        mov     ax,xExt                     ;对于红绿蓝缓存则用像素计数.
@@:     mov     wCount,ax
else
        push    X
        push    Y
        call    ComputeVideoAddress         ;AX:DX <— 视频缓存地址
        mov     lpSource.sel,ax
        mov     lpSource.off,dx
        mov     ax,ds
        mov     es,ax
        mov     di,pBuffer                  ;复制数据的地方.
        push    xExt
        push    1
        call    ComputeLoopParameters       ;BX = 扫描增量, AX = 内循环计数
        cmp     wBitmapType,BM_YUV411       ;如果输出文件是 4:1:1 YUV,
        je      @F                          ;一次做四像素.
        cmp     _wBoardType,BT_YUV411       ;如果读的是 4:1:1 YUV 数据,
        je      @F                          ;一次读四像素.
        mov     ax,xExt                     ;对 RGB 缓存,用像素计数.
@@:     mov     wCount,ax
        lds     si,lpSource
endif
        mov     cx,yExt
read_scan_from_video:
        push    cx                          ;保存外循环计数.
        push    bx                          ;保存扫描增量.
read_pixel_from_video:
        call    npfncVideoToSystem          ;调用选定的格式变换程序,其中一例参看后面.
        dec     wCount
```

```
        jnz         read_pixel_from_video
        pop         bx                          ;扫描增量
        add         si,bx                       ;下一次扫描在视频缓存中的偏移地址
        pop         cx                          ;扫描计数
        loop        read_scan_from_video
        mov         ax,1                        ;成功返回.
exit_ReadImageRect:
cEnd
;**********************************************************************
;程序名        PCV_WriteImageRect                         汇编语言
;造句法    int PCV_WriteImageRect(fpBuffer,fpPalette,X,Y,XExt,YExt,
;              wBitmapType)
;功能          从输入缓存读指定格式图像数据转换成16位RGB或4:1:1 YUV
;              到视频缓存
;输入参数                                 类型/描述
;              fpBuffer                   D 输入缓存的远指针
;              fpPalette                  D 调色板工作区远指针
;              X                          W 图像块的左边座标
;              Y                          W 图像块的顶端座标
;              XExt                       W 块宽度
;              YExt                       W 块高度
;              wBitmapType                W 位图类型
;返回值        无
;调用子程序:   ComputeVideoAddress,ComputeLoopParameters
;              BufferToExtended
;
; On RGB boards each pixel occupies 16 bits, red and blue each occupy
; 5 bits, green is 6 bits.
;
;   Pixel   15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
;   Format  G0  R4  R3  R2  R1  R0  G5  G4  G3  G2  G1  B4  B3  B2  B1  B0
;
; YUV 数据存作 4:1:1 YUV 格式. 每一像素有一 7 位的无符号灰度值,Y.两个色度分量
; U (B-Y) 和 V (R-Y),各是 7 位无符号值. 因为色度取样速率为灰度的四分之一,每四
; 个像素才有一个取样值. 在 IBM MMotion 文件这些位按如下方式分布在三个字中:
;
;   字 0  15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
;         U6  U5  V6  V5  Y06 Y05 Y04 Y03 Y02 Y01 Y00 XX  U4  U3  V4  V3

;   字 1  15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
;         Y16 Y15 Y14 Y13 Y12 Y11 Y10 XX  U2  U1  V2  V1  Y26 Y25 Y24 Y23

;   字 2  15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
```

```
;          Y22 Y21 Y20 XX  U0  XX   V0   XX   Y36  Y35  Y34  Y33  Y32  Y31  Y30   XX

;U 和 V 分别按 127/179 和 127/226 标定来适应 8 位,去掉 LSB.
;
;       Y = 0.299*R + 0.587*G + 0.114*B
;
;       V = R-Y = R - 0.299*R - 0.587*G - 0.114*B
;       V = 0.701*R - 0.587*G - 0.114*B
;       Vmax = 0.701*255 - 0.587*0 - 0.114*0 = 178.76
;       Vmin = 0.701*0 - 0.587*255 - 0.114*255 = -178.76
;       -> V 按 127/179 标定来匹配 8 位.
;
;       U = B-Y = -0.299*R - 0.587*G + B - 0.114*B
;       U = -0.299*R - 0.587*G + 0.866*B
;       Umax = -0.299*0 - 0.587*0 + 0.866*255 = 225.93
;       Umin = -0.299*255 - 0.587*255 + 0.866*0 = -225.93
;       -> U 按 127/226 标定成 8 位.
;
;从 YUV 变为 RGB, U 和 V 首先乘以 226/127 和 179/127. 可用如下公式:
;
;       R = V + Y          B = U + Y
;       G = 1.706*Y - 0.509*R - 0.194*B

ExportProc PCV_WriteImageRect,<BX,CX,DX,SI,DI,DS,ES>
        parmW     pBuffer                    ;输出缓存的偏移地址.
        parmW     pPalette                   ;调色板的偏移地址.
        parmW     X                          ;待写像素块左边
        parmW     Y                          ;            上边.
        parmW     xExt                       ;块宽度
        parmW     yExt                       ;块高度
        parmW     wBitmapType                ;点图类型.

        localV    luma,4
        localW    U
        localW    V
        localW    wCount
        localV    adRGBQuad,16
        localW    npfncSystemToVideo

cBegin
        cld
        mov       bx,wBitmapType             ;计算指针到表.
        and       bx,0Fh
```

```
        mov     ax,_wBoardType
        shl     ax,4
        or      bx,ax
        shl     bx,1
        mov     ax,SystemToVideoTbl[bx]
        mov     npfncSystemToVideo,ax

ifndef DOSLIB
        push    X
        push    Y
        call    ComputeVideoAddress          ;AX:DX <-- 视频缓存地址
        mov     es,ax
        mov     di,dx
        push    xExt
        push    1
        call    ComputeLoopParameters        ;BX = 扫描增量, AX = 内循环计数
else
        mov     ax,ds
        mov     es,ax
        lea     di, ExtMemBfr                ;待写数据地址.
        mov     ax,xExt
        shr     ax,2
endif
        mov     cx,yExt
        cmp     _wBoardType,BT_YUV411
        je      write_have_count
        cmp     wBitmapType,BM_YUV411        ;如果读 4:1:1 YUV 数据,一次四个像素
        je      write_have_count
        mov     ax,xExt                      ;对于 RGB 数据用像素计数.
        cmp     wBitmapType,BM_DIB4D         ;如果读 4 位彩色,一次二像素
        jnz     write_have_count
        shr     ax,1
write_have_count:
        mov     wCount,ax
        mov     si,pBuffer                   ;DS:SI --> 源数据
        mov     dx,pPalette                  ;DS:DX --> 调色板
write_scan_to_video:
        push    cx                           ;保存外循环计数.
        push    bx                           ;保存扫描增量.
write_pixel_to_video:
        call    npfncSystemToVideo           ;读图像数据,变成 16 位
        dec     wCount                       ;RGB, 并写到视频缓存.
        jnz     write_pixel_to_video
```

```
        pop     bx                              ;扫描增量
        add     si,bx                           ;视频缓存下次扫描的偏移地址
        pop     cx                              ;扫描计数
        loop    write_scan_to_video
ifdef DOSLIB
        push    X
        push    Y
        push    xExt
        push    1
        call    BufferToExtended                ;传送数据从缓存到视频括充内存.
        or      ax,ax                           ;不成功退出.
        jz      Exit_WriteImageRect
endif
        mov     ax,1                            ;Return success.
exit_WriteImageRect:
cEnd
```

PCV_ReadImageRect 和 PCV_WriteImageRect 中 call npfncSystemToVideo 随 GRAND-VIDEO 板的种类和图像文件格式而异，以下仅据一例 YUV411UnPackedToRGB8Color：

```
; YUV411UnPackedToRGB8Color
;
; 本子程序读视频缓存地址 DS:SI 4:1:1 YUV 格式的四像素，变为 15 位的 RGB 值，增加对应色的计数. 把未加功色码写到 ES:DI 地址.
;
; 入口:          DS:SI -> 视频缓存四像素组
;                ES:DI -> 目标缓存
; 出口:          DS:SI -> 下一组像素
;                ES:DI -> 目标缓存

public  YUV411UnpackedToRGB8Color
YUV411UnpackedToRGB8Color proc  near
        call    GetLumaChromaUnpacked           ;提取未包装的灰度及色度数据
        xor     bh,bh
        mov     bl,luma[0]
        call    YUVToRGB8Color                  ;变换成 RGB 8 位调色数据
        xor     bh,bh
        mov     bl,luma[1]
        call    YUVToRGB8Color
        xor     bh,bh
        mov     bl,luma[2]
        call    YUVToRGB8Color
        xor     bh,bh
        mov     bl,luma[3]
```

```
        call     YUVToRGB8Color
        ret
YUV411UnPackedToRGB8Color endp
```

它调用的汇编语言子程序有如下几种：

1. GetLumaChromaUnPacked

从视频缓存读四像素，提取四个灰度和两个色度值. 视频数据必须是4:1:1格式. 色度值U,V分别标定成127/226和127/179，因此由他们乘以标定因子的倒数方能得到未标定值. 视频缓存中数据是未包装的（四个像素每组四个字）. 视频存储器中位的组织方式如下：

```
;
;   字 0  15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
;         U6  U5  V6  V5  XX  XX  XX  XX  Y6  Y5  Y4  Y3  Y2  Y1  Y0  XX

;   字 1  15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
;         U4  U3  V4  V3  XX  XX  XX  XX  Y6  Y5  Y4  Y3  Y2  Y1  Y0  XX

;   字 2  15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
;         U2  U1  V2  V1  XX  XX  XX  XX  Y6  Y5  Y4  Y3  Y2  Y1  Y0  XX

;   字 3  15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
;         U0  XX  V0  XX  XX  XX  XX  XX  Y6  Y5  Y4  Y3  Y2  Y1  Y0  XX
;
; 入口：      DS:SI -> 视频缓存中四像素组
; 出口：      U = 226/127 * U 的标定值
;             V = 179/127 * V 的标定值
;             Luma[0:3] = 灰度值
;             DS:SI -> 下一组像素

public  GetLumaChromaUnpacked
GetLumaChromaUnpacked proc      near
        lodsw                           ;AL = 灰度 0, AH = 色度 0 ( 7—4 位)
        and       al,0feh
        mov       luma,al
        mov       al,ah
        and       ax,0c030h             ;屏蔽色度分量.
        shl       al,2
        mov       dx,ax                 ;DH=V 的 D6:D5, DL=U 的 D6:D5
        lodsw
        and       al,0feh
        mov       luma[1],al
        mov       al,ah
```

```
        and     ax,0c030h               ;屏蔽色度分量.
        shr     ah,2
        or      dx,ax                   ;组合色度位 w/MSBs.
        lodsw
        and     al,0feh
        mov     luma[2],al
        mov     al,ah
        and     ax,0c030h               ;屏蔽色度分量.
        shr     ax,2
        shr     ah,2
        or      dx,ax                   ;装配色度位.
        lodsw
        and     al,0feh
        mov     luma[3],al
        mov     al,ah
        and     ax,08020h               ;屏蔽色度分量.
        shr     ax,4
        shr     ah,2
        or      dx,ax                   ;DH =已标定 U , DL = 已标定 V
        call    UnScaleChroma           ;U <— 未标定 U ,V <— 未标定 V
        ret
GetLumaChromaUnpacked endp
```

2. UnScaleChroma

本子程序把两个标定的 8 位色度分量 U 和 V 分别乘以因子 229/127 和 179/127. 结果用局部变量 U 和 V 存储.

```
;
; Entry:        DL = 已标定的 V
;               DH =          U
; Exit:         V  = 未标定的 V
;               U  =          U

public  UnScaleChroma
UnScaleChroma proc      near
        mov     bl,dh                   ;BL = 已标定的 V
        mov     al,dl
        cbw                             ;AX = 已标定的 V (符号扩展)
        mov     dx,361                  ;(361/256 = 179/127)
        imul    dx
        mov     al,ah
        mov     ah,dl
        mov     V,ax                    ;AX = 未标定 V
        mov     al,bl
```

```
        cbw                             ;AX = scaled U (sign extended)
        mov     dx,456                  ;(456/256 = 226/127)
        imul    dx
        mov     al,ah
        mov     ah,dl
        mov     U,ax                    ;AX = unscaled U
        ret
UnScaleChroma endp
```

3. YUVToRGB8Color

```
;
本子程序一组 YUV 值为 16 位 bitRGB 并把结果存于 ES:DI. 输出字格式如下:
;
;   15   14   13   12   11   10   9    8         6    5    4    3    2    1    0
;   0    R4   R3   R2   R1   R0   G4   G3   G2   G1   G0   B4   B3   B2   B1   B0
;
; 他还递增相应颜色的计数器.
;
; 入口:        U = (B-Y)
;              V = (R-Y)
;              BX = 灰度分量
;              ES:DI -> 输出缓存
; 出口:        ES:DI -> 下一像素

public YUVToRGB8Color
YUVToRGB8Color proc near
        call    ConvertYUVToRGB24       ;AH<-绿  AL<-蓝  BL<-红
        call    ConvertRGB24ToRGB16
        and     ax,7FFFh                ;清除绿的最低位.
        stosw                           ;暂存 15 位值.
        mov     bx,ax
        add     bx,bx
        mov     ax,es
        mov     dx,fpWorkSpace.hi
        mov     es,dx
        inc     word ptr es:[bx]        ;增加对应的计数器.
        jo      counter_overflow
        mov     es,ax
        ret
counter_overflow:                       ;计数器溢出
        dec     word ptr es:[bx]
        mov     es,ax
```

```
        ret
YUVToRGB8Color endp
```

4. ConvertYUVToRGB24

```
;
;变一组 YUV 值为 24 位 RGB 值返回结果于 AX 和 BL 中.
;
;        U = B-Y              ->        B = U+Y
;        V = R-Y              ->        R = V+Y
;        Y  = 0.299*R + 0.587*G + 0.114*B          ->
;        G = 1.704*Y - 0.5094*R - 0.1942*B
;
;入口:        U = B-Y
;             V = R-Y
;             BX = Y = 灰度分量
;出口:        AH = 绿
;             AL = 蓝
;             BL = 红
;备注:        U 和 V 是有符号整数.  灰度无符号.

public  ConvertYUVToRGB24
ConvertYUVToRGB24 proc  near
Blue:   mov     ax,bx
        add     ax,U                ;AX = U + Y = 蓝
        jge     @F
        xor     ax,ax
        jmp     Red
@@:     cmp     ax,0ffh
        jle     Red
        mov     ax,0ffh
Red:    mov     dx,bx
        add     dx,V                ;DX = V + Y = 红
        jge     @F
        xor     dx,dx
        jmp     have_red_and_blue
@@:     cmp     dx,0ffh
        jle     have_red_and_blue
        mov     dl,0ffh
have_red_and_blue:
        mov     dh,al
        xchg    dx,bx               ;BH = 蓝   BL = 红   DX = 灰度
Green:  mov     ax,436              ;(436/256) = 1.703
        mul     dx
```

```
        mov     dh,dl
        mov     dl,ah                   ;DX = 1.704 * 灰度
        mov     al,bl                   ;红
        mov     ah,130                  ;(130/256) = 0.5078
        mul     ah
        mov     al,ah
        xor     ah,ah
        sub     dx,ax                   ;DX = 1.704 * 灰度 - 0.508 * 红.
        mov     al,bh                   ;蓝
        mov     ah,50
        mul     ah
        mov     al,ah
        xor     ah,ah
        sub     dx,ax                   ;DX = 1.704 * 灰度 - 0.508 * 红 - 0.195 * 蓝
        jge     check_green_overflow
        xor     dl,dl                   ;设置绿色分量为零.
        jmp     have_rgb
check_green_overflow:
        cmp     dh,0                    ;绿大于 255 吗?
        je      have_rgb                ;不.
        mov     dl,0ffh                 ;是, 定为 255.
have_rgb:
        mov     al,bh
        mov     ah,dl                   ;AH = 绿  AL = 蓝  BL = 红
        ret
ConvertYUVToRGB24 endp
```

5. ConvertRGB24ToRGB16

```
;
; 变 24 位红绿蓝值为 16 位的 RGB 值.
;
; 入口:     AH = 绿  AL = 蓝  BL = 红
; 出口:     AX = 16 位值 (红 D14:D10, 蓝 D4:D0, 绿 D15,D9:D5)


public ConvertRGB24ToRGB16
ConvertRGB24ToRGB16 proc near
        add     al,00000100b            ;蓝截成 5 位.
        jnc     @F
        mov     al,0ffh
@@:     add     ah,00000010b            ;红绿成 6 位
        jnc     @F
        mov     ah,0ffh
```

```
@@:     add     bl,00000100b            ;红截成 5 为 2.
        jnc     @F
        mov     bl,0ffh
@@:     ror     ah,3
        mov     bh,ah
        and     bh,10000000b
        s]hr    ax,3
        and     ah,00000011b
        shr     bl,1
        and     bl,01111100b
        or      bl,bh
        or      ah,bl
        ret
ConvertRGB24ToRGB16 endp
```

以上汇编子程序包含在 CONVERT. ASM 中.

```
/*****************************************************************
* AllocateMemory(wSizeWords)                                     *
*                                                                *
* 分配一指定大小的内存块并返回一远指针                              *
*                                                                *
* 出口: 内存块的远指针                                             *
*          = 0       不能分配内存                                  *
*****************************************************************/
char far * AllocateMemory(WORD wSizeWords)
{
#ifdef DOSLIB
   return(halloc(wSizeWords, sizeof(WORD)));
#else
   if (! (hMemory = GlobalAlloc(GMEM_FIXED | GMEM_ZEROINIT,
     2 * ((DWORD) wSizeWords))))
      return(NULL);
   else return(GlobalLock(hMemory));
#endif
}

/*****************************************************************
* FreeMemory(char far * fpMemory, HANDLE hMem)                   *
*                                                                *
* 释放所述内存块                                                   *
*                                                                *
* Exit: None                                                     *
*****************************************************************/
```

```
void FreeMemory(char far * fpMemory, HANDLE hMem)
{
#ifdef DOSLIB
    hfree((void huge * ) fpMemory);
#else
    GlobalUnlock(hMem);
    GlobalFree(hMem);
#endif
}

;****************************************
;        ExtendedToBuffer

;复制扩充内存块到缓存.
;
;Exported:    No
;Entry:       None
;Modifies:    AX,BX,CX,DX,SI,ES,FLAGS
;****************************************
cProc   ExtendedToBuffer, <NEAR,PUBLIC>,<BX,CX,DX,SI,DI>
        parmW   X
        parmW   Y
        parmW   rectWidth
        parmW   rectHeight
cBegin
        mov     ax,ds
        mov     es,ax
        xor     ax,ax
        lea     si,DefaultGDT
        lea     di,DummyGDT
        mov     cx,size GDT/2
        rep     movsw                       ;GDT 结构初始化

        mov     ax,Y
        mov     bx,X
        call    ComputeVideoAddress         ;图像的物理地址到 DL:AX
        mov     word ptr DummyGDT.GDT_Entry1.GDTEntry_Address,ax
        mov     DummyGDT.GDT_Entry1.GDTEntry_Address[2],dl
        xor     dx,dx
        mov     ax,ds                       ;计算缓存的物理地址
        mov     dl,ah                       ;ds/256
        shr     dl,4                        ;ds/4096
        shl     ax,4                        ;ds * 16
```

```
        add     ax,offset ExtMemBfr             ;ds * 16+offset=缓存物理地址
        adc     dl,0
        mov     word ptr DummyGDT.GDT_Entry2.GDTEntry_Address,ax
        mov     DummyGDT.GDT_Entry2.GDTEntry_Address[2],dl
        mov     cx,rectWidth                    ;每扫描一次的字数
        mov     ax,cx
        add     ax,ax
        dec     ax
        mov     DummyGDT.GDT_Entry1.GDTEntry_Size,ax
        mov     DummyGDT.GDT_Entry2.GDTEntry_Size,ax
        mov     ax,8700h
        mov     si,offset DummyGDT
        int     15h                             ;移动数据块.
ifdef DEBUG
        jz      ETB_Success                     ;扩充内存到缓存成功
        int     3                               ;不成功.
        xor     ax,ax
        push    1
        call    TurnBorder
        jmp     ETB_Exit
endif
ETB_Success:
        mov     ax,1
ETB_Exit:
cEnd

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;       BufferToExtended

;把缓存复制到扩充内存.
;
;Exported:      No
;Entry:         None
;Modifies:      AX,BX,CX,FLAGS
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
cProc   BufferToExtended, <NEAR,PUBLIC>,<BX,CX,DX,SI,DI>
        parmW   X
        parmW   Y
        parmW   rectWidth
        parmW   rectHeight
cBegin
        mov     ax,ds
        mov     es,ax
```

```
        xor     ax,ax
        lea     si,DefaultGDT
        lea     di,DummyGDT
        mov     cx,size GDT/2
        rep     movsw                       ;初始化 GDT 结构.

        mov     ax,Y
        mov     bx,X
        call    ComputeVideoAddress         ;图像物理地址到 DL:AX
        mov     word ptr DummyGDT.GDT_Entry2.GDTEntry_Address,ax
        mov     DummyGDT.GDT_Entry2.GDTEntry_Address[2],dl
        xor     dx,dx
        mov     ax,ds                       ;计算缓存物理地址
        mov     dl,ah
        shr     dl,4
        shl     ax,4
        add     ax,offset ExtMemBfr
        adc     dl,0
        mov     word ptr DummyGDT.GDT_Entry1.GDTEntry_Address,ax
        mov     DummyGDT.GDT_Entry1.GDTEntry_Address[2],dl
        mov     cx,rectWidth                ;每次扫描字数
        mov     ax,cx
        add     ax,ax
        dec     ax
        mov     DummyGDT.GDT_Entry1.GDTEntry_Size,ax
        mov     DummyGDT.GDT_Entry2.GDTEntry_Size,ax
        mov     ax,8700h
        mov     si,offset DummyGDT
        int     15h                         ;移动块.
ifdef DEBUG
        jz      BTE_Success
        int     3                           ;块移动不成功.
        xor     ax,ax
        push    1
        call    TurnBorder
        jmp     BTE_Exit
endif
BTE_Success:
        mov     ax,1

BTE_Exit:
cEnd
```

```
public DebugBreak
DebugBreak proc far
        int         3
        ret
DebugBreak endp

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;               ComputeVideoAddress
;计算图像的物理地址.
;
;Exported:      No
;Entry:         AX = Y
;               BX = X
;Modifies:      AX,BX,DX,FLAGS
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
ComputeVideoAddress proc near
        shl     ax,3                        ;8 * Y
        mov     dl,ah                       ;Y/32
        mov     ah,al                       ;2048 * Y 到 AX,DX 中
        xor     al,al
        add     bx,bx                       ;2 * X
        add     ax,bx                       ;2048 * Y + 2 * X
        mov     bx,_config. wVideoAddr      ;以兆字节为单位的视频缓存区地址
        shl     bl,4                        ;16 * BX
        add     dl,bl                       ;图像数据物理地址
        xor     dh,dh
        ret
ComputeVideoAddress endp

else
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;       ComputeVideoAddress

;计算对应于 X,Y 的视频缓存的地址.
;
;Exported:      No
;Entry:         X,Y
;Modifies:      AX,BX,DX,FLAGS
;Exit:          AX:DX=Address in video buffer corresponding to X,Y
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
cProc   ComputeVideoAddress, <NEAR,PUBLIC>
        parmW       X
        parmW       Y
```

```
cBegin
        mov     ax,Y
        and     ax,01FFh
        shr     ax,5                ;每段 32 次扫描.
        mul     NextSelIncr
        add     ax,FrameBufSel      ;Selector
        mov     bx,Y
        and     bx,01Fh             ;Y 除以 32 的余数
        xchg    bh,bl               ;左移八位.(乘 256)
        shl     bx,3                ;Y * 2048
        mov     dx,X
        shl     dx,1                ;每像素二字节.
        add     dx,bx               ;段内偏移.
cEnd

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;       ComputeLoopParameters

;计算扫描增量和循环计数.
;
;Exported:   No
;Entry:      None
;Modifies:   AX,BX,CX,FLAGS
;Exit:       AX = Loop count
;            BX = Scan Increment
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
cProc   ComputeLoopParameters, <NEAR,PUBLIC>
        parmW   rectWidth
        parmW   rectHeight
cBegin
        mov     bx,rectWidth
        mov     ax,bx
        shr     ax,2                    ;I 内循环计数= 宽度/4
        shl     bx,1                    ;BX = 2 * 宽度
        neg     bx
        add     bx,FRAME_BUF_WIDTH      ;扫描增量 = 2048-(2 * 宽度)
cEnd
sEnd
endif
end
```

7.4 直接寄存器操作子程序

以下介绍的一批子程序都直接对 PCVIDEO 寄存器操作,不是通过口地址对 SAA9001 寄

存器存取就是通过它的通用寄存器对 I2C 总线来对其它芯片寄存器存取。这里有些子程序原来是汇编程序(参看 PCVIDEO. ASM,CONVERT. ASM),为了使读者更容易理解它对寄存器的究竟作了些什么,我们已把它们顺便改成最简短的 BASIC 语言子程序形式。但是子程序的名称基本不变。这里,我们仍旧使用带"_"的变量名,实际上,目前的多数 BASIC (VISUAL BASIC 例外) 语言并不支持,因此这里的子程序在应用时,对变量名还需适当改动(例如用"."代替)。还有一些较底层的子程序是用 C 语言编的。

为了帮助理解和记忆,它们的英文名称大致有如下规律:

用前缀 EN－开头表示允许,赋能某功能,一般只对寄存器某位操作(0 或 1);用前缀 DIS－开头表示不许,停止某功能,一般只对寄存器某位操作(1 或 0);

这样的例程有 EN(DIS)ABLEVIDEO,EN(DIS)ABLECOLORKEY,EN(DIS)ABLEFIELDREPLICATE,EN(DIS)ABLESCALING,EN(DIS)ABLEINTERLACE 等。

加前缀 UN－也代表某功能的否定:例如 UNFREEZEVIDEO,UNSCALVIDEO。

对同一功能的 SET 和 RESET 操作分别表示设定或复原某寄存器的值。GET 则表示参数替代。属于这方面的例子有:SETCOLOR,SETINPUTFORMAT,SETCAPTUREADDRESS,SETVIDEOSOURCE,SET－COLORKEY,SETSKEWFACTOR,SETREGISTER 等。

以 WAIT 开头的子程序是查循 SAA9001 中断屏蔽寄存器的某位来等待视频或 VGA 的水平,垂直回程的开始或结束。为了保证采集和显示窗口图像的完整性,这是必要的。

```
'*****************************************
' 在 VGA 显示器上显示图像                                  BASIC
SUB     PCV_EnableVideo
        out     wPortAddrdx,DW_MODE_CNTL                ' out
        out     wPortAddrdx+1,INP(wportaddrdx+1) OR 3
                                   '读入原先控制字,使能视频数据采集,XXXXXX11B.
End sub
'*****************************************
' 停止在 VGA 上显示.                                       BASIC
SUB     PCV_DisableVideo
        out     wPortAddrdx,DW_MODE_CNTL
        out     wPortAddrdx+1,INP(wportaddrdx+1) AND 252
                                                 '停止显示视频窗,XXXXXX00B.
End Sub
'*****************************************
' 在 VGA 显示器使用采色键显示.                             BASIC
SUB     PCV_EnableColorKey
        out     wPortAddrdx,DW_MODE_CNTL
        out     wPortAddrdx+1,(INP(wportaddrdx+1) and 247) OR 18
                                                     '使用采色键,XXX1XX1XB.
End SUB
'*****************************************
' 在 VGA 显示器上停止使用采色键.                           BASIC
SUB     PCV_DisableColorKey
```

```
        out      wPortAddrdx,DW_MODE_CNTL
        out      wPortAddrdx+1,(INP(wportaddrdx+1) and 237) OR 3
                                                   '停止使用采色键,XXX0XX0XB.
End SUB
'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; PCV_SetColorKey                                                        MASM
```

PC Video 用设定采色键来选用视频或 VGA 输出。

VGA 显存中像素设成此色时将显示视频. 这个子程序假定了内部调色板已被程序设成对于像素组模式(例如 16 色模式)传值时不变。这是一个可靠的假定,因为在像数组方式中,每一像素位由内部调色板传递后,装配成 8 位值后再作为外部调色板的指针。在采色位面模式,内部调色板是 6 位。

```
; Exported:     Yes
; Entry:        iColor
ExportProc PCV_SetColorKey
        parmW   iColor
cBegin
        mov     ax,iColor
        mov     ah,al
        cmp     ah,0FH                      ;色码高于 15 吗?
        ja      sck_set_color_register ;高于,直送 82C9001 内部寄存器
sck_read_int_palette:                       ;读内部 VGA 调色板
        mov     dx,3DAH                     ;输入 VGA 状态寄存器地址
        in      al,dx                       ;清属性双稳.
        mov     dl,0C0H                     ;DX =3C0H= 属性寄存器地址
        mov     al,ah                       ;色码
        out     dx,al                       ;从内部调色板读颜色. 送出索引地址
        inc     dx                          ;读地址
        in      al,dx                       ;在 PC Video 中寄存器所需颜色
        mov     a ] h,al                    ;色码到 AH
        mov     dl,0DAH                     ;DX =3DAH= 输入状态口
        in      al,dx                       ;清属性双稳,指向索引地址
        mov     al,20H                      ;VGA 视频赋能,调色板固定不变.
        mov     dl,0C0H                     ;DX = 属性地址
        out     dx,al                       ;打开视频显示.
sck_set_color_register:
        mov     dx,_config.wPortAddr        ;9001 口地址
        mov     al,TRANS_COLOR              ;色码已在 AH 中!
        out     dx,ax
cEnd
'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
' 内部程序允许帧复制.                                                     BASIC
SUB     EnFieldreplicate
```

```
        out        wPortAddrdx,INTERFACE_CNTL
        out        wPortAddrdx+1,INP(wportaddrdx+1) OR 8
                                                              '打开场复制特征.
        out        wPortAddrdx,AV_MODE_CNTL
        out        wPortAddrdx+1,INP(wportaddrdx) OR 4
                                                              '打开场采集.
End SUB
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
';不许帧复制的内部程序.                                            BASIC
SUB DisableFieldReplicate
        out        wPortAddrdx,INTERFACE_CNTL
        out        wPortAddrdx+1,INP(wportaddrdx+1) AND (NOT 8)
                                                              '关掉场复制特征.
        out        wPortAddrdx,AV_MODE_CNTL
        out        wPortAddrdx+1,INP(wportaddrdx+1) AND (NOT 4)
                                                              '停止场采集.
End SUB
'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
';设定写保护屏闭位.                                                BASIC
; Exported:        Yes
'Entry:            wMask
SUB     SetWriteProtectMask(parmW wMask)
        out        wPortAddr,LUMA_BITMASK
        out        wPortAddrdx+1,wMASK-256 * INT(wMASK/256)
        wWriteProtectMask=wMASK                               '保存新的掩码.
                                                              '编程灰度位掩码.
        out        wPortAddr,LUMA_BITMASK+1
        out        wPortAddrdx+1,INT(wMASK/256) ;编程色度位掩码.
End SUB
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
';设定视频缓存中开始地址来存储俘获视频.                            MASM
';它也称作采集缓存的开始地址.
cProc   SetCaptureAddress,<NEAR,PUBLIC>
        parmW      XX
        parmW      YY
        mov        ax,XX
        shl        ax,1                     ;XX 乘 2.(每像素二字节)
        mov        bx,ax
        xchg       ah,al
        and        ah,0f8h                  ;地址必需是 4 像素边界.
        mov        dx, config. wPortAddr
        mov        al,AB_ADDR_LO            ;程序俘获地址低字节.
        out        dx,ax
```

```
        inc     al                      ;程序中间字节.
        mov     ah,bh                   ;D11—D9 of X
        and     ah,07h                  ;屏闭高位.
        mov     bx,YY
        and     bx,0fffeh
        shl     bx,2                    ;左移 Y 从第三位开始.
        or      ah,bl                   ;用或操作把 Y 置入 D4—D0.
        out     dx,ax                   ;送入寄存器
        inc     al                      ;程序高字节.
        mov     ah,bh                   ;第 D9—D5 位
        out     dx,ax
cEnd
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; SetAcquisitionWindow                                                  MASM
;
; 设定采集窗.
;
; Exported:     No
; Entry:        None
; Modifies:     AX,BX,DX
cProc SetAcquisitionWindow,<NEAR,PUBLIC>
        parmW   X1
        parmW   Y1
        parmW   X2
        parmW   Y2
cBegin
        mov     dx,_config.wPortAddr
        mov     bx,X1
        mov     al,AW_X_START_LO
        mov     ah,bl
        out     dx,ax                   ;采集窗 X 开始低字节
        inc     al
        .errnz  (AW_X_START_HI—AW_X_START_LO—1);本应当等于 0,否则强迫出错
        mov     ah,bh
        out     dx,ax                   ;X 开始高字节
        mov     bx,Y1
        mov     al,AW_Y_START_LO
        mov     ah,bl
        out     dx,ax                   ;Y 开始低
        inc     al
        .errnz  (AW_Y_START_HI—AW_Y_START_LO—1)
        mov     ah,bh
        out     dx,ax                   ;Y 开始高
```

```
        mov     bx,X2
        mov     al,AW_X_END_LO
        mov     ah,bl
        out     dx,ax                       ;X 结束低
        inc     al
        .errnz  (AW_X_END_HI-AW_X_END_LO-1)
        mov     ah,bh
        out     dx,ax                       ;X 结束高
        mov     bx,Y2
        mov     al,AW_Y_END_LO
        mov     ah,bl
        out     dx,ax                       ;Y 结束低
        inc     al
        .errnz (AW_Y_END_HI-AW_Y_END_LO-1)
        mov     ah,bh
        out     dx,ax                       ;Y 结束高
saw_set_video_xy:
        mov     ax,X2
        sub     ax,X1
        inc     ax
        mov     _wVideoWidth,ax             ;新的宽度
        mov     ax,Y2
        sub     ax,Y1
        inc     ax
        mov     _wVideoHeight,ax            ;新的高度
cEnd
;*****************************************
; FreezeVideoUnsynchronized                                     MASM

;停止视频采集.一直等到视频被冻结才返回.
;
; Exported:         No
; Entry:            None
; Modifies:         AX,DX
public FreezeVideoUnsynchronized
FreezeVideoUnsynchronized proc near
        push    cx                          ;保存 CX.
        xor     cx,cx                       ;CX=0 以防无限循环.
fvideo_wait:
        mov     dx,_config.wPortAddr
        mov     al,AV_MODE_CNTL             ;采集模式控制寄存器
        out     dx,al
        inc     dx
```

```
        in        al,dx                    ;视频冻结了吗?
        mov       ah,al                    ;保留所读.
        and       al,NOT 1                 ;清最低位.
        out       dx,al
        test      ah,1
        loopnz    fvideo_wait              ;等到视频冻结
        pop       cx                       ;复原CX.
        ret
FreezeVideoUnsynchronized endp
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
' 允许视频采集.                                                          BASIC
SUB     UnfreezeVideo
        out wPortAddr,AV_MODE_CNTL                               '采集模式控制寄存器
        out wPortAddr+1,inp(wPortAddr+1) or1
    end SUB

,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
';设定水平放大因子.                                                      BASIC
' 放大因子 Zoomfactor(0=1x, 1=2x, 2=4x, 3=8x)
SUB     HorizontalZoom
        DIM ModeParms. ShiftClkStart(4)
        out wPortAddr,DW_WINDOW_ZOOM
        out wportaddr+1,(inp(wportaddr+1) and &HFC)or Zoomfactor
'       设定水平放大因子.
        out config. wPortAddr,SHIFT_CLK_START
        OUT config. wportAddr+1,_ModeParms. ShiftClkStart(Zoomfactor+1)
        ENDSUB
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
' 设定垂直放大因子.                                                       BASIC
' 放大因子 Zoomfactor (0=1x, 1=2x, 2=4x, 3=8x)
SUB VerticalZoom
        out wPortAddr,DW_WINDOW_ZOOM
        out wportaddr,(inp(wportaddr+1) and &Hf3) or(2 * Zoomfactor)
    END SUB     ret
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; 返回视频输入格式(NTSC 或 PAL).                                          MASM
; 返回: AX=CF_NTSC or CF_PAL
ExportProc PCV_GetInputFormat
cBegin
        mov       ax,_config. wCfgFlags
        and       ax,CF_NTSC
cEnd
'* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
；通过译码电路设定视频输入格式为 NTSC 或 PAL 制式的内部程序.                    MASM
；入口：        fNTSC = CF_NSTC or CF_PAL
cProc     SetInputFormat
          parmW     fInputFormat
cBegin
          cmp       fInputFormat,CF_NTSC
          je        NTSC_format
PAL_format:
          and       _config.wCfgFlags,(NOT CF_NTSC)
          push      ADRDMSD                   ;数字多标准译码器
          push      6h                        ;指针
          push      32h                       ;数据
          call      wr_i2c                    ;经 I2C 总线写寄存器
          push      ADRDMSD                   ;数字多标准译码器
          push      8h                        ;指针
          push      38h                       ;数据
          call      wr_i2c
          jmp       sif_set_acquisition_window
NTSC_format:
          or        _config.wCfgFlags,CF_NTSC
          push      ADRDMSD                   ;数字多制式译码器
          push      6                         ;指针
          push      22h                       ;数据
          call      wr_i2c
          push      ADRDMSD
          push      8                         ;指针
          push      77h                       ;数据
          call      wr_i2c
sif_set_acquisition_window:
sif_exit:
cEnd
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
'；获得视频输入源的设定值.                                                      BASIC
'     Input.source.setting (0—3)
SUB GetVideoSource
      Input.source.setting=_config.wVideoSource and 3
End SUB
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
；在译码电路设定输入视频源的内部程序.                                           MASM
;Entry: wSourceSelect
cProc SETVideoSource
parmW wInputSource
cBegin
```

```
        mov     ax,wInputSource
        and     ax,3                        ;只取低二位
        mov     _config._wVideoSource,ax    ;改配置表参数
        cmp     _wBoardType,BT_RGB16        ;板型检查
        je      svs_rgb_board
svs_yuv_board:                              ;YUV 板
        shl     al,3
        or      al,42h
        push    ADR9051                     ;9051 数字多制式译码器
        push    0Ah                         ;指针
        push    ax                          ;数据
        call    wr_i2c                      ;编程寄存器
        jmp     svs_exit                    ;退出
svs_rgb_board:                              ;RGB16 板
        mov     bl,78h
        cmp     _config. wVideoSource,1 ;输入 2 是 SVHS 源.
        jne     @F
        mov     bl,7Dh
@@:     or      al,bl
        push    ADR7191                     ;7191 数字多制式译码器
        push    0Eh                         ;索引
        push    ax                          ;数据
        call    wr_i2c                      ;写 I2C 总线
svs_exit:
        mov     ax,1                        ;返回成功.
cEnd
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
'; 纠错程序用来设定边界色.                                              BASIC
'       wColor = overscan color
SUB TurnBorde
        j=inp( &H3DA)                       ;输入 VGA 状态地址来清属性双稳.
        OUT &H3C0,&H31                      ;由属性地址,打开边界.
        out &H3C0,wColor
End sub
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
'; 停止视频数据的水平和垂直标定.                                        BASIC
SUB DisableScaling
        out wPortAddr,AW_MODE_CNTL                          '存取窗口模式控制寄存器
        out wPortAddr+1,inP(wPortAddr+1) and 243                       '读现在的值.
                                                                  '停止 X 和 Y 标定.
out wPortAddr,AV_MODE_CNTL                                        ,存取视频模式寄存器.
out wPortAddr+1,inp(wPortAddr+1) and 251                            '关掉帧采集模式.
END SUB
```

```
;********************************************
; SetDisplayWindow
;
;送数到寄存器控制视频显示窗的位置和尺度
;改变： AX,BX,CX,DX
;********************************************
cProc   SetDisplayWindow,<NEAR,PUBLIC>
        parmW   XX
        parmW   YY
        parmW   xSize
        parmW   ySize
cBegin
        mov     bx,XX
        or      bx,bx
        jge     @F                      ;BX>=0
        xor     bx,bx                   ;负数改为零.
@@:     add     bx,_ModeParms.DispWinSkewX ;补偿误差失调.
        mov     cx,YY
        or      cx,cx
        jge     @F
        xor     cx,cx                   ;如小于0改为0.
@@:     add     cx,_ModeParms.DispWinSkewY ;补偿误差失调.
        mov     dx,_config.wPortAddr
        mov     al,DW_X_START_LO         ;X显示开始低位寄存器
        mov     ah,bl
        out     dx,ax                   ;送入寄存器
        inc     al                      ;X显示开始高位寄存器
        .errnz  (DW_X_START_HI - DW_X_START_LO -1);万一不等于0,强迫出错.
                                                  ;多此一举!
        mov     ah,bh
        out     dx,ax
        inc     al                      ;Y显示开始低寄存器
        .errnz  (DW_Y_START_LO - DW_X_START_HI -1)
        mov     ah,cl
        out     dx,ax
        inc     al                      ;Y高
        .errnz  (DW_Y_START_HI - DW_Y_START_LO -1)
        mov     ah,ch
        out     dx,ax
        mov     bx,XX
        add     bx,xSize                ;X显示结束位置
        dec     bx
```

```
        add      bx,_ModeParms.DispWinSkewX ;误差失调补偿.
        mov      cx,YY
        add      cx,ySize                   ;Y 显示结束位置
        dec      cx
        add      cx,_ModeParms.DispWinSkewY ;补偿偏移
        inc      al                         ;X 显示结束低寄存器
        .errnz   (DW_X_END_LO - DW_Y_START_HI -1)
        mov      ah,bl
        out      dx,ax
        inc      al                         ;显示 X 结束高位寄存器
        .errnz   (DW_X_END_HI - DW_X_END_LO -1)
        mov      ah,bh
        out      dx,ax
        inc      al                         ;Y 低
        .errnz   (DW_Y_END_LO - DW_X_END_HI -1)
        mov      ah,cl
        out      dx,ax
        inc      al                         ;Y 高
        .errnz   (DW_Y_END_HI - DW_Y_END_LO -1)
        mov      ah,ch
        out      dx,ax
cEnd

;*****************************************************
;       SetDisplayPosition                          MASM

;设定俘获缓存中开始读视频数据的位置
;改变:  AX,BX,DX
```

当 X 和 Y 消阴信号消失后一定位置处开始读视频信号. 视频并不显示而等到经过一定的像素时钟后才开始显示. 这个数值是由编程到显示窗寄存器中的数据来决定. 例如,为了在 X =100 处显示 X=0 的俘获视频信号,X 值应设成 −100. 在 100 个像素时钟后, X 地址计数器;将在俘获缓存中经过零而视频显示的数据是这时的 X 位置的数据.

```
;*****************************************************
cProc   SetDisplayPosition,<NEAR,PUBLIC>
        parmW    XX
        parmW    YY
cBegin
        mov      ax,XX
        neg      ax
        mov      cl,_bHZoomFactor
        mov      bl,cl
        xor      bh,bh
```

```
        add     bx,bx
        inc     cl                      ;每像素两字节
        sar     ax,cl                   ;按水缩所小因子调节—XX.
        sub     ax,_ModeParms. DispAddrSkewX[bx];补偿
        cmp     _wBoardType,BT_RGB16
        je      @F
        and     ax,01feh                ;取偶数屏闭掉不用的位.
@@:     mov     bl,ah
        mov     ah,al
        and     bl,1                    ;保存 XX 的高位.
        mov     dx,_config. wPortAddr
        mov     al,DW_X_PANNING_LO      ;显示 X 平移低位寄存器
        out     dx,ax
        mov     dl,bl

        mov     ax,YY
        neg     ax
        mov     cl,_bVZoomFactor
        add     cl,bVZoomAdjust
        mov     bl,cl
        xor     bh,bh
        add     bx,bx
        sar     ax,cl                   ;按垂直缩小因子调结—YY.
        sub     ax,_ModeParms. DispAddrSkewY[bx]
        xchg    ah,al
        shl     al,4                    ;YY 的最高位.
        and     al,10h
        mov     bl,dl
        or      bl,al                   ;YY 的最高位在 D4, XX 在 D0
        mov     dx,_config. wPortAddr
        mov     al,DW_Y_PANNING_LO      ;显示 YY 平移低位寄存器
        out     dx,ax
        mov     ah,bl
        mov     dx,_config. wPortAddr
        mov     al,DW_PANNING_HI        ;显示平移高位寄存器
        out     dx,ax
cEnd
;*****************************************
;               ScaleVideo
```

设定垂直和水平标定因子并按需要标定。

输入视频时用数字差分技术漏去若干像点来标定。允许标定时,相应轴保留一误差参数。

每次采集一像点时，将增量加到误差参数上，如果溢出，像素(或一扫描行)就存到视频缓存里。否则，就不要它。本子程序计算 X 和 Y 增量并按需要对每方向标定。如果在 Y 方向标定大于 1，允许调节场只采集一场。这样可减少闪烁。标定因子是 6 位小于 1 的分数按下式计算：

```
; x 增量 = 64 * xSize/_wVideoWidth
; y 增量 = 64 * ySize/_wVideoHeight
;小于 1 的分数乘以 64 可作整数运算.
;改变： AX,BX,CX,DX
;****************************************
cProc   ScaleVideo,<NEAR,PUBLIC>
        parmW   xSize
        parmW   ySize

cBegin
sv_compute_x_scale_factor:                      ;计算 X 标定因子
        mov     ax,xSize
        shl     ax,6                            ;64 * xSize
        xor     dx,dx
        mov     bx,_wVideoWidth
        div     bx                              ;64 * xSize/_wVideoWidth
        or      dx,dx                           ;有余数
        je      @F
        inc     ax                              ;商数加一
@@:     mov     cx,ax                           ;先保留在 CX 中
sv_compute_y_scale_factor:                      ;计算 Y 标定因子
        mov     ax,ySize
        shl     ax,6                            ;64 * ySize
        test    _config.wCfgFlags,CF_REPLICATE
        jz      @F                              ;不允许场复制.
        mov     bx,_wVideoHeight                ;检查 Y 可否标定 2 或更多.
        shr     bx,1                            ;视频高度除以 2
        cmp     ySize,bx
        jle     @F                              ;至少要标定下来二倍.
        shr     ax,1                            ;32 * ySize
@@:     xor     dx,dx
        mov     bx,_wVideoHeight
        div     bx                              ;64 * ySize/_wVideoHeight
        or      dx,dx
        je      @F
        inc     ax
@@:     mov     bx,ax                           ;CX = Scale X BX = Scale Y
sv_set_scale_factors:                           ;设定标定因子
        mov     dx,_config.wPortAddr
```

```
        mov     al,AW_MODE_CNTL             ;采集模式控制寄存器
        out     dx,al
        inc     dx
        in      al,dx                       ;读现用值.
        and     al,11110011b                ;采集视频输入帧和偶数场
        cmp     cx,40h                      ; X 标定因子>= 1?
        jge     @F                          ;是,不能标定 X.
        or      al,00000100b                ;允许 X 标定.
@@:     cmp     bx,40h                      ; Y 标定因子 >= 1?
        jge     @F                          ;是, 不能标定 Y.
        or      al,00001000b                ;允许标定 Y
@@:     out dx,al
        dec     dx
        mov     al,AV_MODE_CNTL             ;编程场采集位.
        out     dx,al                       ;采集模式控制寄存器
        inc     dx
        in      al,dx
        and     al,NOT 00000100b            ;清除场采集位'
        cmp     bl,20H                      ;只采集一场吗?
        jg      @F                          ;不, 全采集.
        or      al,00000100b                ;打开场采集模式.
@@:     out     dx,al
        dec     dx
        cmp     cx,40h
        jge     no_x_scale                  ;X 标定因子取为零.
        mov     al,AW_X_SCALING             ;设定 X 标定寄存器
        mov     ah,cl
        out     dx,ax
no_x_scale:
        cmp     bx,40h                      ;Y 标定因子为零吗?
        jge     no_y_scale
        mov     al,AW_Y_SCALING
        mov     ah,bl
        out     dx,ax
        inc     al                          ;编程 Y 场标定寄存器
        .errnz  (AW_Y_SCALING_ODD - AW_Y_SCALING -1)
        out     dx,ax                       ;编程为相同值.
no_y_scale:
cEnd

/*******************************************
* PCV_GetSystemMetrics                     *
* 返回情求信息.          C 语言             *
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
WORD FAR PASCAL PCV_GetSystemMetrics(WORD wMetric)
{
  switch(wMetric) {
    case SM_VIDEOWIDTH:
      return(wVideoWidth);
    case SM_VIDEOHEIGHT:
      return(wVideoHeight);
    case SM_BOARDTYPE:
      return(wBoardType);
    case SM_VERSION:
      return(wVersion);
    case SM_INTERLACE:
      outp(config.wPortAddr,INTERLACE_CNTL);
      return(inp(config.wPortAddr+1) & INTERLACE_ON);
    case SM_REPLICATE:
      return(config.wCfgFlags & REPLICATE_ON);
    case SM_IMAGEWIDTH:
      return(wImageWidth);
    case SM_IMAGEHEIGHT:
      return(wImageHeight);
    case SM_IMAGETYPE:
      return(wImageType);
    default:
      return(0);
    }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* PCV_GetColor                    C 语言                                          *
* 返回所述的颜色设定值.对 RGB 板,只返回亮度和对比度,色度(HUE)是可调的.            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
BYTE FAR PASCAL PCV_GetColor(WORD iColor)
{
BYTE bColorValue;

  if (iColor == HUE)
    bColorValue = config.bColorSettings[HUE]-128;
  else if (wBoardType == BT_RGB16)
    bColorValue = config.bColorSettings[iColor];
  else bColorValue = 4 * config.bColorSettings[iColor];
  return(bColorValue);
}
```

```
/*************************************************************
* PCV_SetColor                 C 语言
* 设定颜色的外部入口 *
*************************************************************/
int FAR PASCAL PCV_SetColor(WORD iColor, BYTE bColorValue)
{
  if (iColor == HUE)
\ bColorValue += 128;
  else if (wBoardType == BT_YUV411)
    bColorValue /= 4;
  SetColor(iColor,(WORD) bColorValue);
}
/*************************************************************
* 设定颜色                C 语言                               *
*************************************************************/
int PASCAL SetColor(WORD iColor, WORD wColorValue)
{
int iAdjust, iValue, i, j, k;

    config.bColorSettings[iColor] = (BYTE) wColorValue;
    if (wBoardType == BT_RGB16) {            /* RGB 板 */
      if (iColor == BRIGHTNESS) {
      for (i=0; i<config.PhixelTable.cPhixelChips; i++) {
          if (config.PhixelTable.aPhixelChips[i].wPhixelAddr == ADR7192) {
           for (j=0; j<config.PhixelTable.aPhixelChips[i].cPhixelRegs; j++) {
             if (config.PhixelTable.aPhixelChips[i].aPhixelRegs[j].rIndex == 0) {
               k = config.PhixelTable.aPhixelChips[i].aPhixelRegs[j].rData;
               break;
               }
             }
           }
        }
      iAdjust = ((int) wColorValue-128);        /* 范围是 -128 到 +127 */
      FreezeVideoUnsynchronized();
      wr_i2c(ADR7192,0,(k & 0xBF));              /* 允许写到便查表(LUT) */
      for (i=0; i<256; i++) {
        iValue = (int) abLUTValues[i] + iAdjust;
        iValue = min(255,iValue);
        iValue = max(0,iValue);
        wr_i2c(ADR7192,1,iValue);
        }
      wr_i2c(ADR7192,0,k);                      /* 停止写 LUT */
```

```
        UnFreezeVideo();
        }
        else if (iColor == HUE) {
        wr_i2c(ADRDMSD,HUE_CONTROL,wColorValue);
        }
        }
    else {                              /* YUV 板 */
        if (iColor == HUE)
    \ wr_i2c(ADRDMSD,HUE_CONTROL,wColorValue);
          else      /* Program one of the registers on the TDA4680. */
    \ wr_i2c(ADR4680,iColor,wColorValue);
        }
}

/*********************************************************************
* PCV_ResetColors                       C 语言                        *
* 复位所有颜色到缺省值.                                                *
*********************************************************************/
void FAR PASCAL PCV_ResetColors()
{
int i, j, k;

  if (wBoardType == BT_RGB16) {             /* RGB board implementation */
    FreezeVideo();
    for (i=0; i<config.PhixelTable.cPhixelChips; i++) {
       if (config.PhixelTable.aPhixelChips[i].wPhixelAddr == ADR7192) {
       for (j=0; j<config.PhixelTable.aPhixelChips[i].cPhixelRegs; j++) {
          if (config.PhixelTable.aPhixelChips[i].aPhixelRegs[j].rIndex == 0) {
            k = config.PhixelTable.aPhixelChips[i].aPhixelRegs[j].rData;
            break;
            }
          }
       }
       }
    wr_i2c(ADR7192,0,(k & 0xBF));          /* 允许写 LUT */
    for (i=0; i<256; i++) {
      abLUTValues[i] = i;
      wr_i2c(ADR7192,1,i);
      }
    wr_i2c(ADR7192,0,k);                /* 停止写 LUT */
    wr_i2c(ADRDMSD,HUE_CONTROL,aDefColorTbl[HUE]);
    config.bColorSettings[BRIGHTNESS] = aDefColorTbl[BRIGHTNESS];
    config.bColorSettings[CONTRAST] = aDefColorTbl[CONTRAST];
```

```
    config.bColorSettings[HUE] = aDefColorTbl[HUE];
    UnFreezeVideo();
    }
  else                                    /* YUV 板 */
    for (i=0; i<NO_COLOR_CONTROLS; i++) {
      if (i == HUE) {
        config.bColorSettings[HUE] = aDefColorTbl[HUE];
        wr_i2c(ADRDMSD,HUE_CONTROL,aDefColorTbl[HUE]);
        }
      else {
        config.bColorSettings[i] = aDefColorTbl[i];
        wr_i2c(ADR4680,i,aDefColorTbl[i]);
        }
      }
}

/*****************************************************************************
* PCV_GetSkewFactor                     C 语言                               *
* 返回所标明的 skew 因子设定值.                                               *
* 入口:    iSkewFactor            对应于:                                    *
*          -------                ---------------------                      *
*              0                      显示窗 X Skew                          *
*              1                             Y Skew                          *
*              2                      显示地址 Skew X                        *
*              3                               Skew Y                        *
*              4                      Shift Clock Start                      *
*              5                      调色板 Skew                            *
*              6                      锁相环分频系数                         *
*****************************************************************************/
WORD FAR PASCAL PCV_GetSkewFactor(int iSkewFactor)
{
  switch (iSkewFactor)
      {
      case SF_DISPWINSKEWX:
        return (config.aMode[iVideoMode].DispWinSkewX
          & aModeRange[iSkewFactor]);
        break;
      case SF_DISPWINSKEWY:
        return (config.aMode[iVideoMode].DispWinSkewY
          & aModeRange[iSkewFactor]);
        break;
      case SF_DISPADDRSKEWX:
        return (config.aMode[iVideoMode].DispAddrSkewX[bHZoomFactor]
```

```
            & aModeRange[iSkewFactor]);
          break;
        case SF_DISPADDRSKEWY:
          return (config.aMode[iVideoMode].DispAddrSkewY[bVZoomFactor]
            & aModeRange[iSkewFactor]);
          break;
        case SF_SHIFTCLOCKSTART:
          return (config.aMode[iVideoMode].ShiftClkStart[bHZoomFactor]
            & aModeRange[iSkewFactor]);
          break;
        case SF_PALETTESKEW:
          return (config.aMode[iVideoMode].PaletteSkew
            & aModeRange[iSkewFactor]);
          break;
        case SF_PLLDIVISOR:
          return (config.aMode[iVideoMode].PLLDivisor
            & aModeRange[iSkewFactor]);
          break;
        default:
          break;
        }
}

/*****************************************************************************
* PCV_SetSkewFactor                    C 语言                                *
*设定所标明的 skew 因子.                                                      *
*入口:同上                                                                   *
*****************************************************************************/
int FAR PASCAL PCV_SetSkewfactor(int iSkewFactor, WORD wValue)
{
int i;
PMODE pMode;
  pMode = (PMODE) &aDefMode[iVideoMode];
  switch (iSkewFactor)
      {
      case SF_DISPWINSKEWX:
          i = (pMode -> DispWinSkewX & ~aModeRange
            [iSkewFactor]) | wValue;
          config.aMode[iVideoMode].DispWinSkewX = ModeParms.
            DispWinSkewX = i;
          break;
      case SF_DISPWINSKEWY:
          i = (pMode -> DispWinSkewY & ~aModeRange[iSkewFactor]) | wValue;
```

```
        config. aMode[iVideoMode]. DispWinSkewY = ModeParms.
          DispWinSkewY = i;
        break;
    case SF_DISPADDRSKEWX:
        i = (pMode -> DispAddrSkewX[bHZoomFactor]
          & ~aModeRange[iSkewFactor]) | wValue;
        config. aMode[iVideoMode]. DispAddrSkewX[bHZoomFactor] =
          ModeParms. DispAddrSkewX[bHZoomFactor] = i;
        break;
    case SF_DISPADDRSKEWY:
        i = (pMode -> DispAddrSkewY[bVZoomFactor] & ~aModeRange
          [iSkewFactor]) | wValue;
        config. aMode[iVideoMode]. DispAddrSkewY[bVZoomFactor] =
            ModeParms. DispAddrSkewY[bVZoomFactor] = i;
        break;
    case SF_SHIFTCLOCKSTART:
        i = (pMode -> ShiftClkStart[bHZoomFactor] & ~aModeRange
          [iSkewFactor]) | wValue;
        config. aMode[iVideoMode]. ShiftClkStart[bHZoomFactor] =
          ModeParms. ShiftClkStart[bHZoomFactor] = i;
        outp(config. wPortAddr,SHIFT_CLK_START);
        outp(config. wPortAddr+1,LOBYTE(i));
        break;
    case SF_PALETTESKEW:
        i = (pMode -> PaletteSkew & ~aModeRange[iSkewFactor]) | wValue;
        config. aMode[iVideoMode]. PaletteSkew = ModeParms. PaletteSkew = i;
        outp(config. wPortAddr,DW_MODE_CNTL);
        i = (inp(config. wPortAddr+1) & 0x3f) | (i << 6);
        outp(config. wPortAddr+1,i);
        break;
    case SF_PLLDIVISOR:
        if (wBoardType == BT_YUV411) {
        i= (pMode -> PLLDivisor & ~aModeRange[iSkewFactor]) | wValue;
        config. aMode[iVideoMode]. PLLDivisor = ModeParms. PLLDivisor = i;
        outp(config. wPortAddr,PLL_DIVISOR_INDEX);   /* High word */
        outp(config. wPortAddr+1,PLL_DIVISOR_HI);
        outp(config. wPortAddr,PLL_DIVISOR_DATA);
        outp(config. wPortAddr+1,HIBYTE(i));
        outp(config. wPortAddr,PLL_DIVISOR_INDEX);          /* Low word */
        outp(config. wPortAddr+1,PLL_DIVISOR_LO);
        outp(config. wPortAddr,PLL_DIVISOR_DATA);
        outp(config. wPortAddr+1,LOBYTE(i));
        }
```

```
            break;
        default:
         break;
        }
  UpdateVideoWindow();                /* 更新视频窗. */
  return(TRUE);
}

/******************************************************************
* PCV_ResetSkewFactors                   C 语言                    *
* 把当前视频模式的 skew 因子复原到缺省值.                            *
******************************************************************/
void FAR PASCAL PCV_ResetSkewFactors()
{
int i;
PMODE pMode;
Y
  pMode = (PMODE) &aDefMode[iVideoMode];
Y
  config.aMode[iVideoMode].DispWinSkewX = ModeParms.DispWinSkewX =
    pMode -> DispWinSkewX;
  config.aMode[iVideoMode].DispWinSkewY = ModeParms.DispWinSkewY =
    pMode -> DispWinSkewY;
  config.aMode[iVideoMode].DispAddrSkewX[bHZoomFactor] =
          ModeParms.DispAddrSkewX[bHZoomFactor] =
      pMode -> DispAddrSkewX[bHZoomFactor];
  config.aMode[iVideoMode].DispAddrSkewY[bVZoomFactor] =
          ModeParms.DispAddrSkewY[bVZoomFactor] =
      pMode -> DispAddrSkewY[bVZoomFactor];
  config.aMode[iVideoMode].ShiftClkStart[bHZoomFactor] =
          ModeParms.ShiftClkStart[bHZoomFactor] =
      pMode -> ShiftClkStart[bHZoomFactor];
  config.aMode[iVideoMode].PaletteSkew = ModeParms.PaletteSkew =
      pMode -> PaletteSkew;
  config.aMode[iVideoMode].PLLDivisor = ModeParms.PLLDivisor =
      pMode -> PLLDivisor;
  /* 复位 Shift Clock Start */
  outp(config.wPortAddr,SHIFT_CLK_START);
  outp(config.wPortAddr+1,LOBYTE(ModeParms.ShiftClkStart[bHZoomFactor]));
  /* 复位调色板 Skew. */
  outp(config.wPortAddr,DW_MODE_CNTL);
  i = (inp(config.wPortAddr+1) & 0x3f) | (ModeParms.PaletteSkew << 6);
  outp(config.wPortAddr+1,i);
```

```
  /* 复原锁相环分频系数. */
  if (wBoardType == BT_YUV411) {
    outp(config.wPortAddr,PLL_DIVISOR_INDEX);          /* High word */
    outp(config.wPortAddr+1,PLL_DIVISOR_HI);
    outp(config.wPortAddr,PLL_DIVISOR_DATA);
    outp(config.wPortAddr+1,HIBYTE(ModeParms.PLLDivisor));
    outp(config.wPortAddr,PLL_DIVISOR_INDEX);          /* Low word */
    outp(config.wPortAddr+1,PLL_DIVISOR_LO);
    outp(config.wPortAddr,PLL_DIVISOR_DATA);
    outp(config.wPortAddr+1,LOBYTE(ModeParms.PLLDivisor));
    }
  UpdateVideoWindow();                    /* 更新视频窗. */
  }

/*******************************************************************
* PCV_SetRegister                C 语言                             *
* 设定 PC Video 寄存器为所标明的值.                                 *
*******************************************************************/
void FAR PASCAL PCV_SetRegister(WORD index, BYTE bValue)
{
BYTE bSavedIndex;

  if (HIBYTE(index) != 0)
    wr_i2c((int) HIBYTE(index),(int) LOBYTE(index),(int) bValue);
  else {
    bSavedIndex = inp(config.wPortAddr);        /* 保存当前指针. */
    outp(config.wPortAddr,(BYTE) index);
    outp(config.wPortAddr+1, bValue);
    outp(config.wPortAddr, bSavedIndex);        /* 复位原来指针 */
    }
}

/*******************************************************************
* PCV_GetRegister                C 语言                             *
* 取 PC Video 寄存器值.                                             *
* 出口:寄存器值                                                     *
*******************************************************************/
BYTE FAR PASCAL PCV_GetRegister(WORD index)
{
BYTE bSavedIndex, bValue;
  if (HIBYTE(index) != 0) {
    bValue = 0xFF;
    }
```

```
  else {
    bSavedIndex = inp(config.wPortAddr);          /* 保存当前指针. */
    outp(config.wPortAddr,(BYTE) index);
    bValue = inp(config.wPortAddr+1);
    outp(config.wPortAddr, bSavedIndex);          /* 复原原先指针 */
    }
  return(bValue);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 * PCV_GetVideoAddress                 C 语言                                            *
 * 提取视频缓存的现地址. 它必需以一兆为边界. 从 1-15.                                   *
int FAR PASCAL PCV_GetVideoAddress(void)
{
  return(config.wVideoAddr);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 * PCV_SetVideoAddress                 C 语言                                          *
 *        设定视频地址,1-15                                                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
int FAR PASCAL PCV_SetVideoAddress(int iVideoAddr)
{
unsigned char i;
      config.wVideoAddr = iVideoAddr = iVideoAddr & 0x0F;
      for (i=0; i < config.PCVideoTable.cPCVideoRegs; i++)
          if (config.PCVideoTable.aPCVideoRegs[i].rIndex == 6) {
            config.PCVideoTable.aPCVideoRegs[i].rData =
                    (config.PCVideoTable.aPCVideoRegs[i].rData
                        & 0xF0) | iVideoAddr;
            break;
            }
      outp(config.wPortAddr,6);
      i = inp(config.wPortAddr+1) & 0xF0;
      outp(config.wPortAddr+1, i | iVideoAddr);
#ifdef DOSLIB
      return(1);
#else                                           /* WINDOWS 库用 */
      FreeSelectors();                  /* 释放旧的选段. */
      return(AllocNInitSelectors());          /* 分配新的选段. */
#endif
}
```

```
/*******************************************
 * PCV_GetPortAddress          C 语言        *
 *返回口地址                                  *
 ******************************************/
int FAR PASCAL PCV_GetPortAddress(void)
{
        return(config.wPortAddr);
}

/*******************************************
 * PCV_SetPortAddress          C 语言        *
 *设定口地址                                  *
 ******************************************/
int FAR PASCAL PCV_SetPortAddress(int iPortAddr)
{
unsigned char i;
        config.wPortAddr=iPortAddr;       /* 保存新的口地址. */
        return(TRUE);
}

/*******************************************
 * PCV_EnableInterlace         C 语言        *
 *允许隔行扫描输出                             *
 ******************************************/
void FAR PASCAL PCV_EnableInterlace(void)
{
int i;
        outp(config.wPortAddr,INTERLACE_CNTL);
        i = inp(config.wPortAddr+1) | INTERLACE_ON;
        outp(config.wPortAddr+1,LOBYTE(i));
}

/*******************************************
 * PCV_DisableInterlace        C 语言        *
 *停止隔行扫描输出*
 ******************************************/
void FAR PASCAL PCV_DisableInterlace(void)
{
int i;
        outp(config.wPortAddr,INTERLACE_CNTL);
        i = inp(config.wPortAddr+1) & ~INTERLACE_ON;
        outp(config.wPortAddr+1,LOBYTE(i));
}
```

以下几个程序是用查讯 SAA9001INTERRUPT_POLL(索引地址 09H)的特定位的方法等到视频图像信号或 VGA 显示信号的垂直或水平回扫的开始或终了.它们都是用执行速度最快的汇编语言编的.

```
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;循环到一帧数据被采集.
;改变:   AX,CX,DX
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
WaitToAcquireField proc near
        mov     dx,_config.wPortAddr
        mov     al,INTERRUPT_POLL
        out     dx,al
        inc     dx
        mov     cx,0ffffh                          ;避免等待太长时间.
@@:     in      al,dx                              ;如果不在垂直回程中,循环.
        test    al,VIDEO_VSYNC
        jne     @F                                 ;如果为 1,退出.
        loop    @B
@@:     call    WaitVideoVerticalRetraceStart
        ret
WaitToAcquireField endp
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;等待视频回扫开始.
;改变:AX,CX,DX
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
public WaitVideoVerticalRetraceStart
WaitVideoVerticalRetraceStart proc near
        mov     dx,_config.wPortAddr
        mov     al,INTERRUPT_POLL
        out     dx,al
        inc     dx
        mov     cx,0ffffh
@@:     in      al,dx                              ;等待垂直同步完结.
        test    al,VIDEO_VSYNC
        loopnz  @B                                 ;如果为 0,出来.
        mov     cx,0ffffh
@@:     in      al,dx                              ;等待垂直同步开始.
        test    al,VIDEO_VSYNC
        loopz   @B                                 ;如果为 1,出来.
        ret
WaitVideoVerticalRetraceStart endp

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
; 等待 VGA 水平回扫结束
; 改变: AX,CX,DX
;****************************************
WaitVGAHorizontalRetraceEnd proc near
        mov     dx,_config.wPortAddr
        mov     al,INTERRUPT_POLL
        out     dx,al
        inc     dx
        mov     cx,0ffffh               ;防止循环太长时间.
@@:     in      al,dx                   ;等待出水平同步区.
        test    al,VGA_HSYNC
        loopnz  @B                      ;如果为 0, 出来.
        mov     cx,0ffffh               ;如果时间太长,不等.
@@:     in      al,dx                   ;等待水平同步的开始.
        test    al,VGA_HSYNC
        loopz   @B                      ;如果为 1, 出来.
ret
WaitVGAHorizontalRetraceEnd endp

;****************************************
; 等待 VGA 垂直回扫开始(只当_wVersion 大于 0 适用).
; 改变: AX,CX,DX
;****************************************
WaitVGAVerticalRetraceStart proc near
        mov     dx,_config.wPortAddr
        m ] ov  al,INTERRUPT_POLL
        out     dx,al
        inc     dx
        mov     cx,0ffffh               ;最长等待时间.
@@:     in      al,dx                   ;等待出垂直同步区.
        test    al,VGA_VSYNC
        loopz   @B                      ;如果为 1, 出来.
        mov     cx,0ffffh               ;最长等待时间.
@@:     in      al,dx                   ;等待垂直同步开始.
        test    al,VGA_VSYNC
        loopnz  @B                      ;如果为 0, 出来.
        ret
WaitVGAVerticalRetraceStart endp
```

以下是用C语言编的经 I2C 总线读写 Phillips 芯片寄存器的子程序. 再下层的子程序。

```
/****************************************/
reverse(register uint data)             /*反转位序,最高位先发送*/
{
```

```
unsigned long rdata = 0;
int i;

  for(i = 0; i < 16; i++){
    rdata |= (data & 1);      /* 不要最低位      */
    data  >>= 1;
    rdata <<= 1;
  }
  return(rdata >> 1);
}                               /* reverse */

/*****************************************/
int i2c_ack()                      /* 从从设备返回应答信号 */
{
register unsigned ack;

  regwrit(I2C_SDA, 1);
  regwrit(I2C_SCL, HICLK);
  ack = regread(I2C_SCL);
  regwrit(I2C_SCL, LOCLK);
  return(ack);
}                                  /* i2c_ack */

/*****************************************/
void i2c_start()                   /* 送开始协议到从设备 */
{
  regwrit(I2C_SDA, 1);
  regwrit(I2C_SCL, HICLK);
  regwrit(I2C_SDA, 0);
  regwrit(I2C_SCL, LOCLK);
} /* i2c_start */

/*****************************************/
void i2c_stop()                 /* 送停止协议到 I2C 从设备,停止协议是当时钟线 */
{                               /* 为高时在数据线上发送正沿 */
  regwrit(I2C_SDA, 0);
  regwrit(I2C_SCL, HICLK);
  regwrit(I2C_SDA, 1);
} /* i2c_stop */

/*****************************************/
void i2c_wdata(uint type, uint data)/*I2C 设备写数据 */
{
```

```
register i;

  data = reverse(data) >> 8;                    /* 最高位在先 */
  if(type == I2C_ADR) i2c_start();              /* 送开始协议 */
  for(i = 0; i < 8; data >>= 1, i++) {          /* 送数据流 */
      regwrit(I2C_SDA, data);
      regwrit(I2C_SCL, HICLK);
      regwrit(I2C_SCL, LOCLK);
  }
  i2c_ack();                                    /* 等待回答 */
}                                               /* i2c_wdata */

/******************************************/
void NEAR PASCAL wr_i2c(int addr, int subaddr, int data)
{
  i2c_wdata(I2C_ADR,addr);                      /* 写地址 */
  i2c_wdata(I2C_SUBADR,subaddr);                /* 写子地址 */
  i2c_wdata(I2C_DAT,data);                      /* 写数据 */
  i2c_stop();
}                                               /* wr_i2c */

/******************************************/
void regwrit(uint type, uint data)              /* 由 I2C 总线写寄存器 */
{
int temp;
  switch (type) {
    case I2C_SCL:                               /* 写时钟线 */
      outp(config.wPortAddr,0x18);              /* SAA9001 的通用寄存器 */
      temp  = inp(config.wPortAddr+1) & 0xfe;   /* 不要原来的最低位 */
    temp |= (data & 1);                         /* 配上时钟数据 */
      outp(config.wPortAddr+1,temp);            /* 送数到通用寄存器 */
      break;
    case I2C_SDA:                               /* 写数据线 */
      outp(config.wPortAddr,0x18);
      temp  = inp(config.wPortAddr+1) & 0xfd;   /* 不要原来的第二位 */
      temp |= ((data & 1) << 1);                /* 配上数据 */
      outp(config.wPortAddr+1,temp);
      break;
  }                                             /* 开关结束 */
}

/******************************************/
int regread(uint type)                          /* 读 I2C 寄存器 */
```

```
{
int temp;
  switch (type) {
    case I2C_SCL:
      outp(config.wPortAddr,0x18);
      temp = inp(config.wPortAddr+1) & 0x01;        /* 读通用寄存器的最低位 */
      break;
    case I2C_SDA:
      outp(config.wPortAddr,0x18);
      temp  = (inp(config.wPortAddr+1) >> 2) & 0x01;
      break;
  } /* switch */
  return(temp);
}
```

第八章　应用编程

在这一章中，我们将介绍 PCVIDEO 的 DOS 应用编程的方法。需要的硬件环境是：

IBM386 以上或兼容机，内存容量 4 到 8 兆字节。VGA 或 TVGA 及多同步显示器。除了上述 PCVIDEO 软件源程序以外，建议软件环境是：UCDOS3.0，MS C/C++ 7.0 和 MASM 6.0. PCVIDEO 原来使用的软件环境是 DOS 3.0，MS C 6.0 和 MASM 5.1. 我们采用的 UCDOS 3.0 代替 DOS 3.0，它不但可支持各种字体和大小的汉字而且与所用的 MS C/C++7.0 及 MASM 6.0 语言兼容。所用的 C 及宏汇编语言版本比 PCVIDEO 原规定的高，并且目前国内正流行，容易得到。

首先是 DOS 库函数 PCVIDEO. LIB 的制作. 下面是 GRAND VIDEO 卡软件中 SDK\DOS 子目中的 PCVIDEO 文件. 它说明了如何在 MS C 6.0 和 MASM 5.1 语言环境下作成与源文件相对应的目标文件(*.OBJ)，再用它们作成 DOS 库函数 PCVIDEO. LIB 的全过程. 其中用" *.OBJ:"开头的行表示作成该目标文件所需的源文件和头(C 语言，*.H)或内包文件(MASM 语言，*.INC)及所在子目。"..\"表示当前目录的父目录. 第二行表示 C 或 MASM 的编译命令及开关参数。

```
pcvideoc.obj: ..\pcvideoc.c ..\include\pcvideo.h ..\include\typedef.h
        cl -c -Gs -Oas -Zpei -W2 -DDOSLIB ..\pcvideoc.c
framegrb.obj: ..\framegrb.c ..\include\pcvideo.h ..\include\typedef.h
        cl -c -Gs -Oas -Zpei -W2 -DDOSLIB ..\framegrb.c
pcvideo.obj: ..\$*.asm ..\include\$*.inc
        masm -v -Mx -DDOSLIB ..\$*;
convert.obj: ..\convert.asm ..\include\pcvideo.inc
        masm -v -Mx -DDOSLIB ..\$*;
pcvideo.lib: pcvideo.obj pcvideoc.obj palette.obj convert.obj framegrb.obj
        del pcvideo.lib
        lib pcvideo.lib +pcvideoc +framegrb +palette +pcvideo +convert;
```

程序 8.1 用 NMAKE 的 PCVIDEO. LIB 库制作文件 PCVIDEO

为了正确使用 PCVIDEO 文件作成 PCVIDEO. LIB，可按如下步骤进行：

1. 设定好 MASM 6.0 和 MS C/C++ 7.0 的共同应用环境，如果环境变量太多而溢出，可暂时删去不用的环境(例如 WINDOWS)；

2. 把 GRANDVIDEO 所附软盘包括上述所有文件及所属子目复制到硬盘中某用户子目中，以便有足够空间编译运行 NMAKE MAKEFILE 命令；

3. 进入该子目中 PCVIDEO 文件所在子目，把 PCVIDEO 改名为 MAKEFILE；

4. 键入 NMAKE 回车后，自动执行 MAKEFILE 文件；

```
pcvideoc.obj: ..\pcvideoc.c ..\include\pcvideo.h ..
```

\include\typedef. h

cl －c －Gs －Oas －Zpei －W2 －DDOSLIB ..\pcvideoc. c

编译成 PCVIDEOC. OBJ 目标文件

5. 用文本编辑器(例如 WPS 或 DOS 6.0 的 EDIT)把 MAAKEFILE 文已经完成任务的前一二句的开头加 # 号(相当于注释),NMAKE 将不管这种语句. 再度执行 NMAKE 时将自动产生 MAKEFILE 中的第二目标文件 FRAMEGRB. OBJ;

6. 反复执行 4 和 5,最终将产生 PCVIDEO. LIB DOS 库文件;

7. 进入 EXAMPLE. C,EXAMPLE. H 及 EXAMPLE 所在子目,把 EXAMPLE 改名为 MAKEFILE 重复 4,5 步骤,直到最终产生出 EXAMPLE. EXE 可执行文件为止;

8. 由于 PCVIDEO. INI 文件也在第二步中被复制到 EXAMPLE. EXE 所在子目. 当 GRANDVIDEO 卡已到位时,可以立即运行 EXAMPLE. EXE. 观察实际效果。

也可改用如下命令行方法在 MASM 6.0 和 MS C/C＋＋ 7.0 环境中,一步一步编译,联结制作成 PCVIDEO. LIB DOS 库函数. 具体步骤如下:

1. 把 PCVIDEO. ASM PCVIDEO. INC 和 CONVERT. ASM 复制到 MASM\BIN 子目中(也可以把 PCVIDEO. INC 复制到 MASM\INCLUDE 子目中),运行 NEW－VARS. BAT 文件,创建运行 MASM 6.0 的必要环境. 进入 MASM\BIN 子目;

2. masm －v －Mx －DDOSLIB pcvideo. asm

masm －v －Mx －DDOSLIB convert. asm 把汇编源程序编译成目标程序;

3. 进入\C700\BIN 子目中把 PCVIDEOC. C,FRAMEGRB. C 复制到\C700\SOURCE 子目中。把 PCVIDEO. H 和 TYPEDEF. H 复制到\C700\INCLUDE 子目中,并且热起动重新设置 C 环境,主要是:SET INCLUDE＝\C700\INCLUDE 和 SET LIB＝\C700\LIB。当然也可以从开始就设置 C 和 MASM 两种语言的共同环境而不需热起动. 只要环境变数不溢出;

4. cl －c －Gs －Oas －Zpei －W2 －DDOSLIB pcvideoc. c

cl －c －Gs －Oas －Zpei －W2 －DDOSLIB framegrb. c

把 C 语言源程序编成相应的目标程序;

5. 把 MASM\BIN 子目中的 PCVIDEO. OBJ,CONVERT. OBJ 和 PCVIDEO 软件所附的 PALETTE. OBJ 复制到 C700\BIN 子目中;

6. lib pcvideo. lib pcvideoc＋framegrb＋palette＋pcvideo＋convert

建成 PCVIDEO. LIB DOS 库函数。

值得注意,以上开关参数大小写敏感。

利用 PCVIDEO. LIB 我们可以用 C 语言编应用程序。PCVIDEO 软件提供的 EXAMPLE. C EXAMPLE. H 和 EXAMPLE 就是一例。它由如下三个文件组成:

EXAMPLE. EXE	DOS 应用程序。它建造一个视频窗口,移动和改变大小,平移活动图像,装入 TARGA 格式图像,放大和移动这图像。它还用不同文件格式存盘及再装入图像。
EXAMPLE. C	DOS 应用示范程序的源程序。
EXAMPLE. H	定义所用常数,子程序和数据结构。

程序 8.2 是已被我们修改加汉字注解,并汉化的 EXAMPLE. C 程序。程序 8.3 是它所用到的内包(头)文件。

```
/*************************************************************************
 * 程序：       EXAMPLE.EXE                                              *
 * 模块：       EXAMPLE..C                                               *
 * 描述：       PC Video DOS 程序样本.                                   *
 * 原作者：     Olive Tao                                                *
 * 版本：       1.5                                                      *
 * 版权     (C) Chips and Technologies, Inc. 1991                        *
 * 汉化及修改： 孙琮琦 中国科学院 沈阳金属研究所   1994                  *
 *************************************************************************/
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <pcvideo.h>                    /* PC Video 库内包文件 */
#include "example.h"                    /* 此样本程序内包文件  */

int fQuit = 0;                          /* 退出到 DOS 的旗 */
int Delay = 1;                          /* 等待（1 秒）*/
int Interval = 15;                      /* 段间隙（5 秒）*/
int wRevision;                          /* 修订版号 */
int wFormat;                            /* 板类型号 */
int iVideoWidth, iVideoHeight;          /* 满屏视频宽度和高度 */
int iColor = 1;                         /* 颜色指针 */
int xExtSize,yExtSize,xStart,yStart;

void main()                             /* 主程序  */
{
      ShowMessage();                    /* 控制键信息 */

      if ( VideoInit() ! = 1 ) {        /* DOS 环境下初始视频 */
        printf("PC Video 不能初始化. \n");
        goto End;
        }
      Wait(Interval);                   /* 等待 5 秒 */

      while( ! fQuit) {                 /* 反复 */
            /* 第 1 段：标定 */
            VideoSize();                /* 显示标定 */
            if (fQuit) goto Exit;
        goto XY(1,1);
        printf(" 改变大小 ");

            Wait(Interval);             /* 等 5 秒 */
```

```
    /* 第 2 段：移动 */
    VideoMove();                    /* Show positioning */
    if (fQuit) goto Exit;
Goto_XY(1,1);
printf("移动 ");
    Wait(Interval);                 /* Wait for 5 Sec. */

    /* Section 3: Panning */
    VideoPan();                     /* Show video panning. */
    if (fQuit) goto Exit;
GOTO_XY(1,1);
printf("平动");
    Wait(Interval);                 /* Wait for 5 Sec. */

    /* Section 4: Zooming */
    if (wRevision != 0){            /* Silicon 版本号 0 不支持放大. */
        Wait(Interval);             /* Wait for 5 Sec. */
Goto_XY(1,1);
printf("放大 ");
        VideoZoom();                /* Show video zooming. */
        }
    if (fQuit) goto Exit;
    Wait(Interval);                 /* Wait for 5 Sec. */

    /* Section 5: 图像存盘和装入 */
    VideoSaveLoad();                /* Show saving & loading image. */
    if (fQuit) goto Exit;
    Wait(Interval);                 /* Wait for 5 Sec. */

    /* Section 6: Testing */
    GOTO_XY(1,1);
    printf("测试其它功能 ");
    VideoTesting();                 /* 测试 */

    iColor = (iColor+1)%15;         /* 循环改变底色 */
    Goto_XY(1,1);
    Background(iColor);
    printf("改变底色 ");
    PCV_SetColorKey(iColor);        /* 复位采色键 */

    }
Exit:                               /* 退出 */
```

```
        PCV_DisableVideo();                          /* 关闭视频窗 */
        PCV_Exit();
End:
        VideoMode(0x3);                              /* 回到正常文本方式. */

}                                                    /* 主程序结束 */


/****************************************************************************
* VideoInit() — 改变视频方式,允许 PV Video.                                 *
****************************************************************************/
int VideoInit()
{
        VideoMode(0x12);                             /* 640x480 16—色图形方式 */
        Background(iColor);                          /* 背景颜色 */
        if (PCV_Initialize() ! = 1)
            return (0);                              /* PC Video 初始化 */
        iVideoWidth = PCV_GetSystemMetrics(SM_VIDEOWIDTH);
        iVideoHeight = PCV_GetSystemMetrics(SM_VIDEOHEIGHT);
        wRevision = PCV_GetSystemMetrics(SM_VERSION);
        wFormat = PCV_GetSystemMetrics(SM_BOARDTYPE);
        PCV_ClearVideoRect(0,0,CAPTURE_BUFFER_WIDTH,
          CAPTURE_BUFFER_HEIGHT);
        PCV_SetColorKey(iColor);                     /* 设定窗口色健 */
        PCV_PanWindow(0, 0);
        PCV_EnableVideo();
        xExtSize = PCV_GetSystemMetrics(SM_VIDEOWIDTH)/2;  /* 缺省视频窗口大小 */
        yExtSize = PCV_GetSystemMetrics(SM_VIDEOHEIGHT)/2;

        xStart = max(0,(SCREENWIDTH—xExtSize)/2);
        yStart = max(0,(SCREENHEIGHT—yExtSize)/2);
        PCV_CreateWindow(xStart,yStart,xExtSize,yExtSize,FITWINDOW);
        return( 1 );                                 /* 成功标志 */
}

/****************************************************************************
* VideoMove() — Show moving video.                                          *
****************************************************************************/
void VideoMove()
{
int i, x, y;
int xDist, yDist;
```

```
    xDist = SCREENWIDTH－xExtSize;                  /* 移动距离 */
    yDist = SCREENHEIGHT－yExtSize;
    x = y = 0;
    for (i=0; i<MOVENO; i++) {
        ControlKey();
        if (fQuit) return;                          /* 退出 */
          PCV_SetWindowPosition(x, y);              /* 对角运动 */
          x = x + xDist/MOVENO;
          y = y + yDist/MOVENO;
          Wait(Delay);
    }
    PCV_SetWindowPosition(xStart,yStart);           /* 回到起始位置 */
}

/*****************************************************************
* VideoSize() — Show sized video.                               *
*****************************************************************/
void VideoSize()
{
int xSize, ySize;
int xMax, yMax;

    xMax = min(iVideoWidth, SCREENWIDTH);
    yMax = min(iVideoHeight, SCREENHEIGHT);
    PCV_SetWindowPosition((SCREENWIDTH－xMax)/2,
      (SCREENHEIGHT－yMax)/2);
    PCV_SetWindowSize(xMax, yMax, FITWINDOW);
    xSize = xMax;
    ySize = yMax;

    while ( (xSize >= 32) && (ySize >= 32 ) ){
        ControlKey();                       /* 扫描键盘输入 */
        if(fQuit) return;
        PCV_CreateWindow((SCREENWIDTH－xSize)/2,
          (SCREENHEIGHT－ySize)/2,xSize,ySize,FITWINDOW);
        xSize = xSize － SIZE_X_INC;
        ySize = ySize － SIZE_Y_INC;
        Wait(Delay/2);
        }

    PCV_SetWindowPosition(xStart, yStart); /* 复原移到 (xStart, yStart) */
    PCV_SetWindowSize(xExtSize, yExtSize, FITWINDOW);/* 窗口大小复原 */
```

```
}

/****************************************************
 * VideoPan() — Show panned video.                  *
 ****************************************************/
void VideoPan()
{
int xRange, yRange;
int xExt, yExt;

    xExt = min(SCREENWIDTH, iVideoWidth);
    yExt = min(SCREENHEIGHT, iVideoHeight);
    PCV_SetWindowPosition((SCREENWIDTH-xExt)/2,
      (SCREENHEIGHT-yExt)/2);
    PCV_SetWindowSize(xExt, yExt, FULLSIZEWINDOW); /* 最大窗. */
    if (wRevision ! = 0)                  /* 硬件先决条件 */
    PCV_EnableFieldReplication();          /* 允许场复制. */
    Wait(Interval);
    PCV_FreezeVideo();                     /* 冻结视频 */
    Wait(Interval);                        /* 显示满屏图像. */
    PCV_SetWindowPosition(xStart, yStart);
    PCV_SetWindowSize(xExtSize, yExtSize, FULLSIZEWINDOW);
    Wait( Delay);
    xRange=iVideoWidth - xExtSize;         /* X 方向平动 */
    yRange=iVideoHeight - yExtSize;        /* y 方向平动 */
    ShowPanning( xRange, yRange);
    if (wRevision ! = 0)
    PCV_DisableFieldReplication();         /* 停止场复制. */
    PCV_SetWindowSize(xExtSize, yExtSize, FITWINDOW); /* 匹配窗口. */
    Wait( Delay);
    PCV_PanWindow(0, 0);
    PCV_UnFreezeVideo();                   /* 活动视频 */
}

/****************************************************
 * VideoZoom() — 显示放大视频图像.                    *
 ****************************************************/
void VideoZoom()
{
int i, iZoomMax, iFactor;
int xRange, yRange, xImage, yImage;
int iExpand = 1;                           /* 扩展平动范围. */
WORD wReturnCode;
```

```
        iZoomMax = 2;
        PCV_SetWindowSize(0, 0, FULLSIZEWINDOW); /* 最大窗. */
        PCV_FreezeVideo();                        /* 冻结采集的视频. */
        if ((wReturnCode = PCV_LoadImageRect("HVFLY.TGA",0,0)) < 1)
        PCV_TurnBorder(wReturnCode+1);
        xImage = PCV_GetSystemMetrics(SM_IMAGEWIDTH);      /* 图像宽度 */
        yImage = PCV_GetSystemMetrics(SM_IMAGEHEIGHT);    /* 图像高度 */
        PCV_SetWindowPosition((SCREENWIDTH-xImage)/2,
          (SCREENHEIGHT-yImage)/2);
        PCV_SetWindowSize(xImage, yImage, FULLSIZEWINDOW); /* 最大窗. */
        if (wFormat == BT_YUV411 ) iZoomMax = 1;        /* 411 板,最多放大 2 */
        Wait( Interval);
        PCV_SetWindowPosition(0, 0);
        for ( i=1; i<= iZoomMax; i++ ){
                ControlKey();
                if(fQuit) return;
                iFactor = i;
                PCV_HorizontalZoom(iFactor);
                PCV_VerticalZoom(iFactor);          /* 视频被放大 2. */
                iExpand = 2 * iExpand;
                xRange = xImage * iExpand;
                yRange = yImage * iExpand;
                ShowZooming(xRange, yRange);
                Wait(Interval); /* 等待 2 秒. */
                }
        PCV_SetWindowPosition((SCREENWIDTH-xImage)/2,(SCREENHEIGHT-yImage)/2);
        PCV_SetWindowSize(xImage, yImage, FULLSIZEWINDOW); /* 最大窗. */
        PCV_HorizontalZoom(0);
        PCV_VerticalZoom(0);                 /* 复原. */
        Wait(Interval);
        PCV_UnFreezeVideo();                 /* 允许视频采集. */
        PCV_SetWindowPosition(xStart, yStart);
        PCV_SetWindowSize(xExtSize, yExtSize, FITWINDOW);
}

/******************************************************************
* VideoSaveLoad() — 视频图像的存盘和重装.                          *
******************************************************************/
void VideoSaveLoad()
{

        VideoSave(BM_YUV411, "example.mmp");
```

```
        if(fQuit) return;
        Wait(Interval);
        VideoLoad(BM_YUV411, "example.mmp");
        if(fQuit) return;
        Wait(Interval);
        VideoSave(BM_DIB24, "example.bmp");
        if(fQuit) return;
        Wait(Interval);
        VideoLoad(BM_DIB24, "example.bmp");
        if(fQuit) return;
        Wait(Interval);
        VideoSave(BM_DIB8P, "example.bmp");
        if(fQuit) return;
        Wait(Interval);
        VideoLoad(BM_DIB8P, "example.bmp");
        if(fQuit) return;
        Wait(Interval);
        VideoSave(BM_DIB8G, "example.bmp");
        if(fQuit) return;
        Wait(Interval);
        VideoLoad(BM_DIB8G, "example.bmp");
        if(fQuit) return;
        Wait(Interval);
        VideoSave(BM_TRG24, "example.tga");
        if(fQuit) return;
        Wait(Interval);
        VideoLoad(BM_TRG24, "example.tga");
}

/******************************************
* VideoSave() — 视频图像存盘.             *
******************************************/
void VideoSave( int fType, char * filename)
{
WORD wReturnCode;

        PCV_FreezeVideo();
        Goto_XY( 20, 5);

        switch (fType) {
                case BM_YUV411:
                        printf( " 存盘为 IBM MMotion 格式文件 " );
                        if ((wReturnCode = PCV_SaveImageRect
```

```
        ( filename, 0, 0, xExtSize, yExtSize, fType, 0)) < 1)
        PCV_TurnBorder(wReturnCode+1);
      break;

    case BM_DIB24:
      printf( " 存盘为 24—位 Windows DIB 格式文件 " );
      if ((wReturnCode = PCV_SaveImageRect
          ( filename, 0, 0, xExtSize, yExtSize, fType, 0)) < 1)
        PCV_TurnBorder(wReturnCode+1);
      break;

    case BM_DIB8G:
      printf( " 存盘为 8—位 gray—scale Windows DIB 格式文件. " );
      if ((wReturnCode = PCV_SaveImageRect
          (filename, 0, 0, xExtSize, yExtSize, fType, 0)) < 1)
        PCV_TurnBorder(wReturnCode+1);
      break;

    case BM_DIB8P:
      printf( " 存盘为 8—为彩色 Windows DIB 格式文件. " );
      if ((wReturnCode = PCV_SaveImageRect
          (filename, 0, 0, xExtSize, yExtSize, fType, 0)) < 1)
        PCV_TurnBorder(wReturnCode+1);
      break;

    case BM_TRG24:
      printf( " 存盘为 24—位 Targa 格式文件 " );
      if ((wReturnCode = PCV_SaveImageRect
          (filename, 0, 0, xExtSize, yExtSize, fType, 0)) < 1)
        PCV_TurnBorder(wReturnCode+1);
      break;

    case BM_TRG16:
      printf( " 存盘为 16—位 Targa 格式文件 " );
      if ((wReturnCode = PCV_SaveImageRect(filename,
          0, 0, xExtSize, yExtSize, BM_TRG16, 0)) < 1)
        PCV_TurnBorder(wReturnCode+1);
      break;

    default:
      printf(" 不能存这种文件格式");
}
```

```
        ControlKey();
        PCV_UnFreezeVideo();
        Wait(Interval);
}
/*************************************************************
* VideoLoad() — 从磁盘装入视频图像.                               *
*************************************************************/
void VideoLoad( int fType, char * filename)
{
WORD wReturnCode;

        PCV_FreezeVideo();
        Goto_XY( 20, 5);

        switch (fType) {
                case BM_YUV411:
                        printf( " 装入 IBM MMotion 文件 " );
                        if ((wReturnCode = PCV_LoadImageRect
                                (filename , 0, 0)) < 1)
                          PCV_TurnBorder(wReturnCode+1);
                        break;

                case BM_DIB24:
                        printf( " 装入 24—位 Windows DIB 文件 " );
                        if ((wReturnCode = PCV_LoadImageRect
                                (filename, 0, 0)) < 1)
                          PCV_TurnBorder(wReturnCode+1);
                        break;

                case BM_DIB8G:
                        printf( " 装入 8—位 gray—scale Windows DIB 文件 " );
                        if ((wReturnCode = PCV_LoadImageRect
                                (filename, 0, 0)) < 1)
                          PCV_TurnBorder(wReturnCode+1);
                        break;

                case BM_DIB8P:
                        printf( " 装入 8—位彩色 Windows DIB 文件 " );
                        if ((wReturnCode = PCV_LoadImageRect
                                (filename, 0, 0)) < 1)
                          PCV_TurnBorder(wReturnCode+1);
                 break;
```

```
        case BM_TRG24:
            printf( "装入 24－位 Targa 文件 " );
            if ((wReturnCode = PCV_LoadImageRect
                (filename, 0, 0)) < 1)
              PCV_TurnBorder(wReturnCode+1);
            break;

        case BM_TRG16:
            printf( "装入 16－位 Targa 文件 " );
            if ((wReturnCode = PCV_LoadImageRect
                (filename, 0, 0)) < 1)
              PCV_TurnBorder(wReturnCode+1);
            break;

        default:
            printf(" 不能装入这种格式文件 ");
    }

    ControlKey();
    Wait(Interval);
    PCV_UnFreezeVideo();
    Wait(Interval);
}

/*****************************************
 * Wait() — 等待观看 PC Video.              *
 *****************************************/
void Wait(int delay)       /* 等待时间单位 1/10 秒 */
{
register int i, k;

    k = delay;
    if (delay > 0) {
    for (i = 0; i < k; i++) {
        while (! (inp(0x3da) & 0x08));
        while ((inp(0x3da) & 0x08));
        }
    }
} /* Wait */

/*****************************************
 * VideoTesting() — 测试各种 PC 功能.        *
 *****************************************/
```

```
void VideoTesting()
{
BYTE      bOriginalColor[NO_COLOR_CONTROLS];
int       iOriginalSkewFactor[NO_SKEW_FACTORS];
int       i, iPortAddr, iVideoAddr;
    GOTO_XY(1,1);
    printf("复位彩色寄存器 ");
    PCV_ResetColors();                          /* 复位 7 彩色寄存器到缺省值. */
    wait(Interval);
    GOTO_XY(1,1);
    printf("获取彩色寄存器的值");
    for(i = 0; i < NO_COLOR_CONTROLS; i++)
        bOriginalColor[i] = PCV_GetColor(i);
    wait(Interval);
    GOTO_XY(1,1);
    printf("设定彩色寄存器的值");
    for(i = 0; i < NO_COLOR_CONTROLS; i++)   /* 设定彩色寄存器. */
        PCV_SetColor(i,bOriginalColor[i]);
    wait(Interval);
    GOTO_XY(1,1);
    printf("复位相移因子值");
    PCV_ResetSkewFactors();                     /* 复位相移 skew 因子到缺省值. */
    wait(Interval);
    GOTO_XY(1,1);
    printf("读取相移因子值");
    for (i = 0; i < NO_SKEW_FACTORS; i++)
        iOriginalSkewFactor[i] = PCV_GetSkewFactor(i);
    GOTO_XY(1,1);
    printf("设定相移因子值");

    for (i = 0; i < NO_SKEW_FACTORS; i++)
        PCV_SetSkewFactor(i,iOriginalSkewFactor[i]);
    wait(Interval);
    iPortAddr = PCV_GetPortAddress();          /* 获取口地址. */
    PCV_SetPortAddress(iPortAddr);             /* 设定口地址. */

    iVideoAddr = PCV_GetVideoAddress();        /* 获取视频地址(以兆字节为单位).
    PCV_SetVideoAddress(iVideoAddr);           /* 设定视频地址. */

    /* Silicon 版本 0: 不支持的测试项目. */
    if ( wRevision ! = 0 ){
        PCV_EnableInterlace();                 /* 允许隔行扫描. */
        PCV_DisableInterlace();                /* 逐行扫描. */
```

```
        PCV_SetWindowSize(xExtSize, yExtSize, FULLSIZEWINDOW); /* 最大窗. */
        PCV_EnableFieldReplication();          /* 允许场复制. */
        PCV_DisableFieldReplication();         /* 停止场复制. */
        PCV_SetWindowSize(xExtSize, yExtSize, FITWINDOW); /* 匹配窗口. */
    }
}

/************************************************************************
 * ShowZooming() — Show zoomed image.                                   *
 ************************************************************************/
void ShowZooming(int xRange, int yRange)
{
int xExt, yExt;

    if((xRange > SCREENWIDTH) || (yRange > SCREENHEIGHT)) {
        xExt = min( SCREENWIDTH, xRange );
        yExt = min( SCREENHEIGHT, yRange );
        PCV_SetWindowPosition(0, 0);
        PCV_SetWindowSize(xExt, yExt, FULLSIZEWINDOW);
        ShowPanning(xRange-xExt, yRange-yExt);
        }
    else PCV_SetWindowSize(xRange, yRange, FULLSIZEWINDOW);
}

/************************************************************************
 * ShowPanning() — 采集框不变,显示其中一部分.                               *
 ************************************************************************/
void ShowPanning(int xRange, int yRange)
{
int i, xPan, yPan;

    xPan = yPan = 0;
    for ( i=0; i<= xRange; i+= XINC){
        ControlKey();
        if(fQuit) return;
        xPan=i;
        PCV_PanWindow(xPan, yPan);             /* 水平向右平移. */
        Wait(Delay/2);
        }
    for ( i=0; i<= yRange; i+= YINC){
        ControlKey();
        if(fQuit) return;
        yPan=i;
```

```
        PCV_PanWindow(xPan, yPan);                    /* 垂直向下平移. */
        Wait(Delay/2);
        }

    for ( i=xRange; i>=0 ; i-= XINC){
        ControlKey();
        if(fQuit) return;
        xPan=i;
        PCV_PanWindow(xPan, yPan);                    /* 水平向左平移. */
        Wait(Delay/2);
        }
    for ( i=yRange; i>=0 ; i-= YINC){
         ControlKey();
         if(fQuit) return;
         yPan=i;
         PCV_PanWindow(xPan, yPan);                   /* 垂直向上平移. */
         Wait(Delay/2);
        }
}

/**************************************************************************
* ShowMessage()                                                           *
**************************************************************************/

void ShowMessage()
{
int iScanCode;

    printf(" \n");
    printf(" \n");
    printf("PC Video Dos 库样例程序, 版 1.0\n");
    printf(" (c) Chips & Technologies, July 1991\n");
    printf(" \n");
    printf(" 程序执行过程中,可以使用如下控制键 .\n");
    printf(" \n");
    printf(" f 快 \n");
    printf(" s 慢 \n");
    printf(" p 暂停 \n");
    printf(" c 继续 \n");
    printf(" ESC/q 退出 \n");
    printf(" \n");
    printf(" %c 按任意键开始示范 \n",7);
    iScanCode = ScanKeybd();
```

```
    if ((iScanCode == ESC_KEY) || (iScanCode == Q_KEY)) fQuit = 1;
}

/****************************************
* ScanKeybd() — 读键盘不回显，返回扩充键代码           *
****************************************/
int ScanKeybd()
{
int ext_key_code;
int a,b;

    b = 0;
    a = getch();
    if (a == 0) {
      b = 0x100;
      a = getch();
      }
    ext_key_code = b+(a & 0xff);
    return(ext_key_code);
}

/****************************************
* ControlKey() — 检查是否按过控制=键.                 *
****************************************/
void ControlKey()
{
    if(kbhit()){
      switch( ScanKeybd() ) {
    if(kbhit()){
      switch( ScanKeybd() ) {
        case ESC_KEY:                    /* 退出程序. */
        case Q_KEY:
            fQuit = 1;
            break;

          case P_KEY:                    /* 暂停程序. */
            while ( ScanKeybd() ! = C_KEY )
            break;

          case F_KEY:                    /* 加速移动. */
            if (Delay > 0 )
                Delay --;
            break;
```

```
            case S_KEY:                             /* 减慢运动. */
              if (Delay < 65534 )
              Delay = Delay++;
              break;

          default: return;
          }
        }
      return;
}

/*************************************************************************
* VideoMode() — 设定视频显示方式 (BIOS 功能 10h, 子功能).                *
*************************************************************************/
int VideoMode(int mode_no)
{
union REGS reg86;
int res86;

    reg86.h.ah = 0;
    reg86.h.al = mode_no;
    res86 = int86(0x10,&reg86,&reg86);
    return(res86);
}

/*************************************************************************
* Goto_XY() — 放显示文本的光标.                                           *
*************************************************************************/
void Goto_XY(int x, int y)
{
union REGS r;

      r.h.dl = x;
      r.h.dh = y;
      r.h.bh = 0;
      r.h.ah = 2;
      int86(0x10, &r, &r);
}

/*************************************************************************
* Background() — 设定底色为蓝.                                            *
*************************************************************************/
```

```
void Background(int color)
{
char far *screenPtr;
uint iOffset;
uint i, j, x, y;

      outpw(0x3ce,color << 8);                    /* 设定色玛到置位/重置寄存器 */
      outp(0x3ce,5);                              /* 图形模式寄存器 */
      outp(0x3cf,0x03);                           /* 写模式 3 */

      for( i=0; i< SCREENHEIGHT; i++){ y = i;
          for( j=0; j< SCREENWIDTH; j++){
              x = j;
              iOffset = SCREENWIDTH * y+ x;   /* VGA 显存偏移地址 */
              screenPtr = (char far *) 0xa0000000L+iOffset;
              *screenPtr = 0xff;                  /* 掩码 */
          }
      }
      outp(0x3cf,0);                              /* 写模式 0 */
}
```

程序 8.2 EXAMPLE.C

```
/*****************************************************************
* Program: EXAMPLE.EXE                                            *
* Module: EXAMPLE.H                                               *
* Description: PC Video DOS 库样本程序的内包文件.                  *
* Version: 1.5                                                    *
* Copyright (C) Chips and Technologies, Inc. 1991                 *
*****************************************************************/

#define ESC_KEY             0x1b      /* ESC 键对应的扩展码 */
#define Q_KEY               0x71      /* Q 键对应的扩展码 */
#define P_KEY               0x70      /* P 键对应的扩展码 */
#define C_KEY               0x63      /* C 键对应的扩展码 */
#define F_KEY               0x66      /* F 键对应的扩展码 */
#define S_KEY               0x73      /* S 键对应的扩展码 */
#define SCREENWIDTH         640       /* 屏宽 */
#define SCREENHEIGHT        480       /* 屏高 */
#define FITWINDOW           1         /* 匹配视频旗 */
#define FULLSIZEWINDOW      0
#define MOVENO              100       /* 移动次数 */
#define XINC                4         /* X 增量 */
```

```
#define YINC                3          /* Y 增量 */
#define SIZE_X_INC          4
#define SIZE_Y_INC          3

int VideoInit(void);
void VideoMove(void);
void VideoSize(void);
void VideoPan(void);
void VideoZoom(void);
void VideoSaveLoad(void);
void VideoSave(int, char * );
void VideoLoad(int, char * );
int VideoMode(int);
void Wait(int);
void VideoTesting(void);
void ShowZooming(int, int);
void ShowPanning(int, int);
void ShowMessage(void);
int ScanKeybd(void);
void ControlKey(void);
void Goto_XY(int, int);
void Background(int);
```

程序 8.3 EXAMPLE.C 的内包文件 EXAMPLE.H。

仿照 3 4 相同步骤可以获得 EXAMPLE.OBJ. 然后把它与 PCVIDEO.LIB ,MS C/C++ 7.0 自带库函数 OLDNAMES.LIB 和 SLIBCE.LIB 联结成 EXAMPLE.EXE 可执行文件. 联结命令格式是:

link /NOD/CO example+pcvideo.lib,,,oldnames slibce

这里/NOD 表示不需纠错(DEBUG),oldname.LIB 和 slibce.LIB 是 C 语言中两个所用到的库函数。

在 UCDOS 3.0 环境下 EXAMPLE.C 中可以进一步改进,UCDOS 本身提供了以像素座标为单位,在任意位置写各种大小和不同字体汉字的功能。这可以用来画汉字说明的动画。画线和画圆弧的命令则可用来画彩色的可变圆框或多角形框,利用色键功能 PCV_SETCOLORKEY 设定色键为相同颜色,就能实现本书第一章曾经介绍过的 DEMO.BAT 利用便笺文件所完成的同类工作。PCV_DEM.EXE 命令解释器可识别的以字母 G 开头的各种命令都属于在 VGA 上画一组有动画效果图形的命令,它们与 GRANDVIDIO 硬件无关,在许多书中已有这种用C 语言作图的优秀程序,在作者的"TVGA 卡应用开发入门"一书中,也已分析了 TANGO 3.16 软件中用汇编语言快速画圆,画直线的子程序.这里不再介绍。

对于C 语言不很熟练,但有 BASIC 编程经验的读者,原则上可以把以上 *.OBJ 文件作成 QUICK BASIC,MS BASIC 7.1 或 VISUAL BASIC FOR DOS 的快速库.以 MS BASIC 7.1 为例,建快速库的 PCVIDEO.QLB 和编译成独立可执行文件所必需的库 PCVIDEO.LIB 的方法

如下：

```
LIB pcvideo.lib +pcvideoc+pcvideo+framegrb+convert+qbx.lib
LINK/Q/NOD/CO pcvideo.lib,pcvideo.qlb,,qbxqlb.lib
```

需要利用 MS BASIC 7.1 中鼠标，菜单，窗口甚至数据库的读者，应该在以上包括 LIB 的一行中添加上 general，mouse，menu，window，uiasm 等项。

当快速库 PCVIDEO.QLB 建成后，我们可以仿照 EXAMPLE.C 的格式，编成相应的 BASIC 源程序 EXAMPLE.BAS(参看程序 10.3，在改编中，我们也已增加了少量显示汉字的功能！)，直接在 UCDOS3.0 的汉字环境中 BASIC 的集成开发环境中运行它. 由于 UCDOS 有在 BASIC 程序中制作各种大小，字体的汉字的功能，MS BASIC 7.1 又能支持鼠标，菜单，窗口，绘图和数据库的功能，不难在它们的基础上编制成任意实用的大型软件. 除矩形窗口以外，这里可以利用 BASIC 的方便的作图功能及 PCVIDEO 的色键来实现其它形状的活动窗口。

```
DECLARE FUNCTION videoinit% ()
DECLARE SUB waitt (interval%)
REM $INCLUDE: 'PCVIDEO.QBI'
REM     主程序
      GOSUB ShowMessage                          ' * Control key info * '
      IF (videoinit% <> 1) THEN                  '在 UCDOS 环境下初始化 GRAND VIDEO 卡
      PRINT "PC Video 初始化不成功!"
        GOTO Endex
      END IF
      iVideoWidth = pcvgetsystemmetrics%(sm.videowidth)'
      iVideoHeight = pcvgetsystemmetrics%(SM.VIDEOHEIGHT)'
      wRevision = pcvgetsystemmetrics%(SM.VERSION)'
      wFormat = pcvgetsystemmetrics%(SM.BOARDTYPE)'
      value% = pcvclearvideorect%(0, 0,
        CAPTUREBUFFERWIDTH, CAPTUREBUFFERHEIGHT)'
      value% = pcvsetcolorkey%(iColor)
      value% = pcvpanwindow%(0, 0)               '设定窗口色键
      value% = pcvenablevideo%'
      xExtSize = pcvgetsystemmetrics%(sm.videowidth) 'Default video size *'
      yExtSize = pcvgetsystemmetrics%(SM.VIDEOHEIGHT)
      xStart =abs (SCREENWIDTH - xExtSize) / 2'
      yStart = abs (SCREENHEIGHT - yExtSize) / 2'
      value% = PCVCreateWindow%(xStart, yStart, xExtSize,yExtSize, FITWINDOW$)'
      waitt interval%
      DO WHILE fQuit <> 1
            GOSUB VideoSize                      '标定
            gosub controlkey
            IF (fQuit=1) THEN goto Endex '
            waitt interval%                      ' * Wait for 5 Sec * '
```

```
        GOSUB VideoMove'
        gosub controlkey                            '移动
        IF fQuit=1 THEN goto Endex '
        waitt interval%'                            ' * Wait for 5 Sec. * '
        GOSUB VideoPan'
        gosub controlkey                            '平移
        IF fQuit=1 THEN goto Endex'
        waitt interval%'                            ' * Wait for 5 Sec. * '
        IF wRevision <> 0 THEN                      '版本 0 不支持放大
            waitt interval%'                        ' * Wait for 5 Sec. * '
        gosub VideoZoom'                            '放大
            END IF
        gosub controlkey
        IF fQuit=1 THEN goto Endex
        waitt interval%'                            ' * Wait for 5 Sec. * '
        ftype = bmyuv411
        filename$ = "example.mmp"
        GOSUB videosave                             '存盘
        IF INKEY$ = "Q" THEN RETURN'
        ftype = bmyuv411
        filename$ = "example.mmp"'
        GOSUB VideoLoad                             '装入图像
        IF INKEY$ = "Q" THEN RETURN'
        ftype = bmdib24
        filename$ = "example.bmp"
        GOSUB videosave
        gosub controlkey
        IF fQuit=1 THEN goto Endex'
        waitt interval%'                            ' * Wait for 5 Sec. * '
        gosub VideoTesting'                         '测试
        iColor = iColor + 16 * int((1-icolor+1)/16)'改变底色
        color iColor '
        value% = pcvsetcolorkey%(iColor)            '复原底色
        LOOP
Endex:
        value% = pcvdisablevideo%                   '关闭视窗
        value% = pcvexit%'
        SCREEN 3, 0                                 '回到文本模式
    END

REM 以下子程序:
ControlKey:
    C% = ASC(INKEY$)
```

```
            IF C% = ESCKEY OR C% = QKEY THEN
              fQuit = 1'
            ELSEIF C% = PKEY THEN '          ' * Pause the program.  * '
            DO WHILE INKEY$ <> "C": LOOP
            ELSEIF C% = FKEY THEN '          ' * Fast motion.  * '
              IF (Delay > 0) THEN Delay = Delay - 1'
            ELSEIF (Delay < 65534) THEN Delay = Delay + 1 '
        END IF
RETURN

ShowMessage:
        PRINT
        PRINT
        PRINT "PCVideo Dos BASIC 快速库调用示范程序, 1.0 版"
        PRINT " (c) 沈阳金属研究所,1994 年 4 月"
        PRINT "孙琮琦"
        PRINT " 当程序执行时,可按如下键改变程序执行进程 ."
        PRINT
        PRINT " f 快 "
        PRINT " s 慢 "
        PRINT " p 暂停 "
        PRINT " c 继续 "
        PRINT " ESC 或 q 回到 DOS 环境 "
        PRINT
        PRINT " 按任意键开始示范."
        iScanCode = ASC(INKEY$) '
        IF iScanCode = ESCKEY THEN
          iScanCode = QKEY
            fQuit = 1
        END IF
RETURN

DEFINT I, W-Y
ShowPanning:
        xPan = 0: yPan = 0'
        FOR i = 0 TO xrange STEP XINC
            IF INKEY$ = "Q" THEN RETURN'
            xPan = i'
            value% = pcvpanwindow%(xPan, yPan)          '水平平移
            waitt Delay / 2
          NEXT
        FOR i = 0 TO yRange STEP YINC
            IF INKEY$ = "Q" THEN RETURN'
```

```
            yPan = i'
            value% = pcvpanwindow%(xPan, yPan)              '垂直平移
            waitt Delay / 2
         NEXT
      FOR i = xrange TO 0 STEP -XINC
            IF INKEY$ = "Q" THEN RETURN'
            xPan = i'
            value% = pcvpanwindow%(xPan, yPan)              '水平平移
            waitt Delay / 2
         NEXT
      FOR i = yRange TO 0 STEP -YINC
            IF INKEY$ = "Q" THEN RETURN'
            yPan = i'
            value% = pcvpanwindow%(xPan, yPan)'             '垂直平移
         CALL waitt(Delay / 2) '
         NEXT

RETURN

ShowZooming:
      IF ((xrange > SCREENWIDTH) AND (yRange > SCREENHEIGHT)) THEN
         xExt = SCREENWIDTH
         IF xExt > xrange THEN xExt = xrange'
         yExt = SCREENHEIGHT
         IF yExt > yRange THEN yExt = yRange
         value% = pcvsetwindowposition%(0, 0)'
         value% = pcvsetwindowsize%(xExt, yExt, FULLSIZEWINDOW$)'
         x=xrange-xext
         y=yrange-yext
         gosub showpanning
      ELSE
      value% = pcvsetwindowsize%(xrange, yRange, FULLSIZEWINDOW$)'
      END IF
RETURN

DEFINT I, W-Y
VideoLoad:
      value% = pcvfreezevideo%'
      LOCATE 20, 5'
      ON ftype GOTO myuv411, mdib24, mdib8g, mdib8p, mtrg24, mtrg16
myuv411:
           PRINT " 装入 IBM MMotion 文件. "
                 GOSUB PASSTRTC
```

```
        wreturncode=pcvloadimagerect%(aoff%,aSEG%, 0, 0)
    if wreturncode< 1 THEN
        PCVturnborder wreturncode + 1
    END IF '
    GOTO default
mdib24:
    PRINT " 装入 24 位 Windows DIB 文件."
      gosub passtrtc
      wreturncode = pcvloadimagerect%(aoff%, aSEG%, 0, 0)
      if wreturncode < 1 THEN
        PCVturnborder wreturncode + 1'
      END IF
      GOTO default
mdib8g:
      PRINT " 装入 8—位灰度级别 Windows DIB 文件." '
      gosub passtrtc
      wreturncode = pcvloadimagerect%(aoff%, aSEG%, 0, 0)
      if wreturncode < 1 THEN
        PCVturnborder wreturncode + 1'
      END IF
      GOTO default
mdib8p:
      PRINT " 装入 8 位彩色 Windows DIB 文件. " '
      gosub passtrtc
      wreturncode = pcvloadimagerect%(aoff%, aSEG%, 0, 0)
      if wreturncode < 1 THEN
        PCVturnborder wreturncode + 1'
      END IF
      GOTO default
mtrg24:
      PRINT " 装入 24 位 Targa 文件. " '
      gosub passtrtc
      wreturncode = pcvloadimagerect%(aoff%, aSEG%, 0, 0)
      if wreturncode < 1 THEN
        PCVturnborder wreturncode + 1

      END IF                  GOTO default
mtrg16:
      PRINT " 装入 16 位 Targa 文件. "
      gosub passtrtc
      wreturncode = pcvloadimagerect%(aoff%, aSEG%, 0, 0)
      if wreturncode < 1 THEN
        PCVturnborder wreturncode + 1'
```

```
            END IF
default:
            PRINT " 不能装入这种文件格式. "
            CALL waitt(interval%)
            value% = pcvunfreezevideo%'
            waitt interval%
RETURN

VideoMove:
      xDist = SCREENWIDTH - xExtSize                          '运动距离
      yDist = SCREENHEIGHT - yExtSize'
      x = 0: y = 0'
      FOR i = 0 TO MOVENO - 1
            IF INKEY$ = "Q" THEN RETURN'
            value% = pcvsetwindowposition%(x, y)                '对角运动
            x = x + xDist ' MOVENO'
            y = y + yDist ' MOVENO'
            waitt Delay'
      NEXT
      value% = pcvsetwindowposition%(xStart, yStart)
RETURN

VideoPan:
      xExt = SCREENWIDTH
      if xext> iVideoWidth then xext=ivideowidth
      yExt = SCREENHEIGHT
      if Yext> iVideoHeight then Yext=ivedeoheight
      value% = pcvsetwindowposition%((SCREENWIDTH - xExt) / 2,
          (SCREENHEIGHT - yExt) / 2)'
      value% = pcvsetwindowsize%(xExt, yExt, FULLSIZEWINDOW$)'最大窗
      IF (wRevision <> 0) THEN
        pcvenablefieldreplication                               '允许场复制
        CALL waitt(interval%)
        value% = pcvfreezevideo%                                '冻结视频
        CALL waitt(interval%)                                   '显示满屏像
      END IF
      value% = pcvsetwindowposition%(xStart, yStart)'
      value% = pcvsetwindowsize%(xExtSize, yExtSize, FULLSIZEWINDOW$)'
      CALL waitt(Delay)
      xrange = iVideoWidth - xExtSize                           'X 平移范围
      yRange = iVideoHeight - yExtSize                          'y 平移范围
      gosub ShowPanning
      IF (wRevision <> 0) THEN
```

```
        pcvdisablefieldreplication                             '不许场复制
        value% = pcvsetwindowsize%(xExtSize, yExtSize, FITWINDOW$)' 匹配窗口
        CALL waitt(Delay)
      END IF
      value% = pcvpanwindow%(0, 0)'
      value% = pcvunfreezevideo%                               '采集视频
RETURN

videosave:
      value% = pcvfreezevideo%'
      LOCATE 20, 5'
      ON ftype GOTO bmyuv411, bmdib24, bmdib8g, bmdib8p, bmtrg24, bmtrg16
bmyuv411:
              PRINT " 存 IBM MMotion 文件 "
              gosub passtrtc
              wreturncode = pcvsaveimagerect%
                (aoff%, aSEG%, 0, 0, xExtSize, yExtSize, ftype, 0)
              if wreturncode < 1 THEN
                PCVturnborder wreturncode + 1'
              END IF
              GOTO defaul
bmdib24:
              PRINT " 存 24 位 Windows DIB 文件 "
              gosub passtrtc
              wreturncode = pcvsaveimagerect%
                (aoff%, aSEG%, 0, 0, xExtSize, yExtSize, ftype, 0)
              if wreturncode < 1 THEN
                PCVturnborder wreturncode + 1'
              END IF
              GOTO defaul
bmdib8g:
              PRINT " 存 8-位灰度级别 Windows DIB 文件. "
              gosub passtrtc
              wreturncode = pcvsaveimagerect%
                (aoff%, aSEG%, 0, 0, xExtSize, yExtSize, ftype, 0)
              if wreturncode < 1 THEN
                PCVturnborder wreturncode + 1'
              END IF
              GOTO defaul
bmdib8p:
              PRINT " 存 8 位彩色 Windows DIB 文件. "
              gosub passtrtc
              wreturncode = pcvsaveimagerect%
```

```
        (aoff%, aSEG%, 0, 0, xExtSize, yExtSize, ftype, 0)
      if wreturncode < 1 THEN
        PCVturnborder wreturncode + 1
      END IF
      GOTO defaul
bmtrg24:
      PRINT " 存 24 位 Targa 文件. "
      gosub passtrtc
      wreturncode = pcvsaveimagerect%
        (aoff%, aSEG%, 0, 0, xExtSize, yExtSize, ftype, 0)
      if wreturncode < 1 THEN
        PCVturnborder wreturncode + 1
      END IF
      GOTO defaul
bmtrg16:
      PRINT " 存 16-位 Targa 文件. "
      gosub passtrtc
      wreturncode = pcvsaveimagerect%
        (aoff%, aSEG%, 0, 0, xExtSize, yExtSize, bmtrg16, 0)
      if wreturncode < 1 THEN
        PCVturnborder wreturncode + 1
      END IF
defaul:
      PRINT " 不能存这种文件格式. "
      value% = pcvunfreezevideo%'
      CALL waitt(interval%)
RETURN

VideoSize:
    xMax = iVideoWidth
    IF xMax > SCREENWIDTH THEN xMax = SCREENWIDTH
    yMax = iVideoHeight
    IF yMax > SCREENHEIGHT THEN yMax = SCREENHEIGHT
    value% = pcvsetwindowposition%(SCREENWIDTH - xMax / 2,
      (SCREENHEIGHT - yMax) / 2)'
    value% = pcvsetwindowsize%(xMax, yMax, FITWINDOW$)
    xSize = xMax
    ySize = yMax
    IF (xSize >= 32) AND (ySize >= 32) THEN
        IF INKEY$ = "Q" THEN RETURN
        value% = PCVCreateWindow%((SCREENWIDTH - xSize) / 2,
          (SCREENHEIGHT - ySize) / 2, xSize, ySize, FITWINDOW$)'
```

```
            xSize = xSize - SIZEXINC
            ySize = ySize - SIZEYINC'
            waitt Delay / 2
        END IF
          value% = pcvsetwindowposition%(xStart, yStart) '移到(xStart, yStart) *'
          value% = pcvsetwindowsize%(xExtSize, yExtSize, FITWINDOW$)
RETURN

VideoTesting:
dim iOriginalSkewFactor(255),bOriginalColor(255)
        pcvresetcolors                                      '恢复 7 种彩色寄存器到缺省值
        FOR i = 0 TO NOCOLORCONTROLS - 1                          '提取彩色寄存器值.
            bOriginalColor(i) =ASC( pcvgetcolor$(i))
        NEXT
        FOR i = 0 TO NOCOLORCONTROLS - 1                    '设定彩色寄存器.
            a$ = pcvsetcolor$(i, bOriginalColor(i))
            NEXT
        value%=PCVResetSkewFactor%                          '复原失调因子到确省值.
        FOR i = 0 TO NOSKEWFACTORS - 1
            iOriginalSkewFactor(i) = pcvgetskewfactor%(i)
            NEXT
        FOR i = 0 TO NOSKEWFACTORS - 1
        value% = pcvsetskewfactor%(i, iOriginalSkewFactor(i))
            NEXT
        iPortAddr = pcvgetportaddress                       '取口地址.
        value% = pcvsetportaddress%(iPortAddr)              '设定口地址
        iVideoAddr% = pcvgetvideoaddress%                   '取视频地址(单位为兆字节).
        value% = pcvsetvideoaddress%(iVideoAddr%)           '设定视频地址
' * Silicon 版 0:不支持的测试项目
        IF wRevision <> 0 THEN
            pcvenableinterlace                              '允许隔行扫描
            pcvdisableinterlace                             '不许隔行扫描
          value% = pcvsetwindowsize%(xExtSize, yExtSize, FULLSIZEWINDOW$)
                                                            '最大窗
         pcvenablefieldreplication                          '允许场复制
         pcvdisablefieldreplication                         '不许场复制
         value% = pcvsetwindowsize%(xExtSize, yExtSize, FITWINDOW$)
                                                            '匹配窗
        END IF
RETURN

VideoZoom:
        iExpand = 1                                         '扩大平移因子范围
```

```
iZoomMax = 2
value% = pcvsetwindowsize%(0, 0, FULLSIZEWINDOW$)'最大窗
value% = pcvfreezevideo%                        '停止采集视频
filename$ ="HVFLY.TGZ"
gosub passtrtc
wreturncode = pcvloadimagerect%(aoff%,aseg%, 0, 0)
if wreturncode < 1 THEN
PCVturnborder wreturncode + 1
END IF
xImage = pcvgetsystemmetrics%(SM.IMAGEWIDTH)  '图像宽度
yImage = pcvgetsystemmetrics%(SM.IMAGEHEIGHT) '高度
value% = pcvsetwindowposition%((SCREENWIDTH - xImage) / 2,
    (SCREENHEIGHT - yImage) / 2)'
value% = pcvsetwindowsize%(xImage, yImage, FULLSIZEWINDOW$)'最大窗
IF (wFormat = BTYUV411) AND (iZoomMax = 1) THEN '对 411 板,放大不到 2
CALL waitt(interval%)
END IF
value% = pcvsetwindowposition%(0, 0)
FOR i = 1 TO iZoomMax
      gosub ControlKey
      IF fQuit THEN RETURN
      iFactor = i
      pcvhorizontalzoom iFactor
      pcvverticalzoom iFactor                    '视频放大 2.
      iExpand = 2 * iExpand
      xrange = xImage * iExpand
      yRange = yImage * iExpand
      gosub ShowZooming
      CALL waitt(interval%)                      '等 2 秒
NEXT
value% = pcvsetwindowposition%((SCREENWIDTH - xImage) / 2,
    (SCREENHEIGHT - yImage) / 2)'
value% = pcvsetwindowsize%(xImage, yImage,FULLSIZEWINDOW$)'
                                                  '* Full size window. *
pcvhorizontalzoom 0
pcvverticalzoom 0
CALL waitt(interval%)
value% = pcvunfreezevideo%                       '视频采集
value% = pcvsetwindowposition%(xStart, yStart)
value% = pcvsetwindowsize%(xExtSize, yExtSize, FITWINDOW$)
RETURN

DEFSNG S
```

```
FUNCTION videoinit%
SCREEN 12, iColor                                   '640x480 16 色
IF (pcvinitialize <> 1) THEN
    videoinit% = 0'                                 '初始化 PC Video
END IF
END FUNCTION

SUB waitt (interval%)
FOR isub = 1 TO 30000
FOR j = 1 TO interval%
NEXT
NEXT
END SUB

PASSTRTC:
fILENAME$ = FILENAME$ + " "
IF 2 * (LEN(FILENAME$) / 2) - LEN(FILENAME$) <> 0 THEN
FILENAME$ = FILENAME$ + " "
END IF
addr = SADD(FILENAME$)
DEF SEG = VARSEG(FILENAME$)
FOR i = 0 TO LEN(FILENAME$) - 2 STEP 2
IL = ASC(MID$(FILENAME$, i + 1, 1))
IH = ASC(MID$(FILENAME$, i + 2, 1))
ichar%(i / 2 + 1) = IH * 256 + IL
NEXT
aoff%=varptr(ichar%(1))
aseg%=varseg(ichar%(1))
return
```

程序 8.4 PCVIDEO BASIC 示范程序 EXAMPLE. BAS

运行 EXAMPLE. BAS 的方法之一是：

QBX/RUN EXAMPLE/L PCVIDEO. QLB.

也可以在 UCDOS，BASIC 带 PCVIDEO. QLB 环境下直接编译 EXAMPLE. BAS 成独立可执行文件的 EXAMPLE. EXE。

在调用 PCVIDEO 快速库函数时，由于 MS BASIC 7.1 不支持带"_"的变量或子程序名，因此原来子程序名和变量名中"_"在快速库中已被省略. 所有这些改动及其它工作已由内包文件 PCVIDEO. QBI 实现. 读者编应用程序时，只需在源程序头加上一句：

REM $ INCLUDE：'PCVIDEO. QBI'

即可按一般调用子程序方法来调用它们。内包文件 PCVIDEO. QBI 参看程序 8.5。用 DECLARE 语句来说明外部非 BASIC 语言编写的子程序，ALIAS 表示改名为后续字符串. BY-

VAL 表示后续参数传值。其它内容同 PCVIDEO. INC 和 EXAMPLE. H。

```
REM 此文件相当于 PCVIDEO. INC 或 PCVIDEO. H 和 TYPEDEF. H
DECLARE FUNCTION PCVCreateWindow% ALIAS "PCV_CreateWindow"
    (BYVAL a%, BYVAL b%, BYVAL C%, BYVAL d%, ch$)
DECLARE FUNCTION pcvsetwindowposition% ALIAS "PCV_SetWindowPosition"
    (BYVAL a%, BYVAL b%)
DECLARE FUNCTION pcvsetwindowsize% ALIAS "PCV_SetWindowSize"
    (BYVAL a%, BYVAL b%, ch$)
DECLARE FUNCTION pcvsetvideoscaling% ALIAS "PCV_SetVideoScaling"
    (BYVAL a%, BYVAL b%, ch$)
DECLARE FUNCTION pcvsetcaptureaddress% ALIAS "PCV_SetCaptureAddress"
    (BYVAL a%, BYVAL b%)
DECLARE FUNCTION pcvsetdisplaywindow% ALIAS "PCV_SetDisplayWindow"
    (BYVAL a%, BYVAL b%, BYVAL C%, BYVAL d%)
DECLARE FUNCTION pcvsetaccquisitionwindow% ALIAS "PCV_SetAcquisitionWindow"
    (BYVAL a%, BYVAL b%, BYVAL C%, BYVAL d%)
DECLARE FUNCTION pcvpanwindow% ALIAS "PCV_PanWindow"
    (BYVAL ja%, BYVAL jb%)
DECLARE FUNCTION pcvsetcolorkey% ALIAS "PCV_SetColorKey" (BYVAL colkey%)
DECLARE FUNCTION pcvfreezevideo% ALIAS "PCV_FreezeVideo"
DECLARE FUNCTION pcvunfreezevideo% ALIAS "PCV_UnFreezeVideo"
DECLARE FUNCTION pcvsetwriteprotestmask% ALIAS
    "PCV_SetWriteProtectMask" (BYVAL a%)
DECLARE FUNCTION pcvenablevideo% ALIAS "PCV_EnableVideo"
DECLARE FUNCTION pcvdisablevideo% ALIAS "PCV_DisableVideo"
DECLARE FUNCTION pcvsaveimagerect% ALIAS "PCV_SaveImageRect"
    (byval aoff%,byval aseg%, BYVAL a%, BYVAL b%, BYVAL C%,
    BYVAL d%, BYVAL e%, BYVAL F%)
DECLARE FUNCTION pcvloadimagerect% ALIAS "PCV_LoadImageRect"
    (byval aoff%,byval aseg%, BYVAL a%, BYVAL b%)
DECLARE FUNCTION pcvreadimagerect% ALIAS "PCV_ReadImageRect"
    (PSTR&, BYVAL a%, BYVAL b%, BYVAL C%, BYVAL d%, BYVAL e%, ch$)
DECLARE FUNCTION pcvwriteimagerect% ALIAS "PCV_WriteImageRect"
    (PSTR&, QSTR&, BYVAL a%, BYVAL b%, BYVAL C%, BYVAL d%, BYVAL e%)
DECLARE FUNCTION pcvgetinputformat% ALIAS "PCV_GetInputFormat"
DECLARE FUNCTION pcvsetinputformat% ALIAS "PCV_SetInputFormat" (BYVAL a%)
DECLARE FUNCTION pcvsetcolor$ ALIAS "PCV_SetColor" (BYVAL a%,byval B%)
DECLARE FUNCTION pcvgetcolor$ ALIAS "PCV_GetColor" (BYVAL a%)
DECLARE SUB pcvresetcolors ALIAS "PCV_ResetColors"
DECLARE FUNCTION pcvclearvideorect% ALIAS "PCV_ClearVideoRect"
    (BYVAL a%, BYVAL b%, BYVAL C%, BYVAL d%)
DECLARE FUNCTION pcvloadconfigration% ALIAS "PCV_LoadConfiguration"
```

```
DECLARE FUNCTION pcvsaveconfigration% ALIAS "PCV_SaveConfiguration"
DECLARE FUNCTION pcvgetportaddress% ALIAS "PCV_GetPortAddress"
DECLARE FUNCTION pcvgetvideoaddress% ALIAS "PCV_GetVideoAddress"
DECLARE FUNCTION pcvgetvideosource% ALIAS "PCV_GetVideoSource"
DECLARE FUNCTION pcvsetvideosource% ALIAS "PCV_SetVideoSource" (BYVAL a%)
DECLARE FUNCTION pcvgetskewfactor% ALIAS "PCV_GetSkewFactor" (BYVAL a%)
DECLARE FUNCTION pcvresetskewfactor% ALIAS "PCV_ResetSkewFactors"
DECLARE SUB pcvenableinterlace ALIAS "PCV_EnableInterlace"
DECLARE SUB pcvdisableinterlace ALIAS "PCV_DisableInterlace"
DECLARE SUB pcvhorizontalzoom ALIAS "PCV_HorizontalZoom" (BYVAL a%)
DECLARE SUB pcvverticalzoom ALIAS "PCV_VerticalZoom" (BYVAL a%)
DECLARE SUB pcvenablefieldreplication ALIAS "PCV_EnableFieldReplication"
DECLARE SUB pcvdisablefieldreplication ALIAS "PCV_DisableFieldReplication"
DECLARE SUB pcvsetregister ALIAS "PCV_SetRegister" (BYVAL a%, ch$)
DECLARE FUNCTION pcvgetregister ALIAS "PCV_GetRegister" (BYVAL a%)
DECLARE SUB PCVturnborder ALIAS "PCV_TurnBorder" (BYVAL a%)
DECLARE FUNCTION pcvinitialize% ALIAS "PCV_Initialize"
DECLARE SUB pcvwaitvgaretrace ALIAS "PCV_WaitVGARetrace"
DECLARE SUB pcvenablecolorkey ALIAS "PCV_EnableColorKey"
DECLARE SUB pcvdisablecolorkey ALIAS "PCV_DisableColorKey"
DECLARE FUNCTION pcvgetprofileint% ALIAS "PCV_GetProfileInt"
    (LPSTR&, MPSTR&)
DECLARE FUNCTION pcvgetprofilestring% ALIAS "PCV_GetProfileString"
    (LPSTR&, MPSTR&, NPSTR&, BYVAL a%)
DECLARE FUNCTION pcvsetportaddress% ALIAS "PCV_SetPortAddress" (BYVAL a%)
DECLARE FUNCTION pcvsetskewfactor% ALIAS "PCV_SetSkewFactor"
    (BYVAL a%, BYVAL b%)
DECLARE FUNCTION pcvexit% ALIAS "PCV_Exit"
DECLARE FUNCTION pcvsetvideoaddress% ALIAS
    "PCV_SetVideoAddress" (BYVAL a%)
DECLARE FUNCTION pcvgetsystemmetrics% ALIAS
    "PCV_GetSystemMetrics" (BYVAL a%)
dim iOriginalSkewFactor(255),bOriginalColor(255),ichar%(20)

CAPTURE.BUFFER.WIDTH = 1024          '* Dimensions of the capture *'
CAPTURE.BUFFER.HEIGHT = 512          '* buffer in pixels *'

LOCLK = 0
HICLK = 1

PLL.DIVISOR.INDEX = &H13
PLL.DIVISOR.DATA = &H12
PLL.DIVISOR.LO = 0
```

```
PLL.DIVISOR.HI = 1
AV.MODE.CNTL = &H20               '* Acquisition video mode register *'
AW.MODE.CNTL = &H21               '* Acquisition window control register *'
NON.MULTIPLEXED = &H10            '* Input is not multiplexed. *'
ACQUIRE.FIELD = 4
SHIFT.CLK.START = &H4C
DW.MODE.CNTL = &H40

INTERLACE.CNTL = &H50
INTERLACE.ON = 1
REPLICATE.ON = 8

I2C.ADR = 0
I2C.SUBADR = 1
I2C.DAT = 2

I2C.SCL = 0
I2C.SDA = 1

'* Color adjustments *'
BRIGHTNESS = 0
SATURATION = 1
CONTRAST = 2
HUE = 3
RED = 4
GREEN = 5
BLUE = 6

'* Phixel chip addresses *'
ADRDMSD = &H8A       '* Digital multistandard decoder *'
ADR7191 = &H8A       '* Square pixel digital multistandard decoder 4:2:2 *'
ADR7192 = &HE0       '* Color space converter *'
ADR9051 = &H8A       '* Digital multistandard decoder 4:1:1 *'
ADR4680 = &H88       '* Color processor *'

HUE.CONTROL = 7

'* System metrics *'
sm.videowidth = 0
SM.VIDEOHEIGHT = 1
SM.BOARDTYPE = 2
SM.VERSION = 3
SM.INTERLACE = 4
SM.REPLICATE = 5
SM.IMAGEWIDTH = 6
SM.IMAGEHEIGHT = 7
```

```
SM.IMAGETYPE = 8

'* Mode flags *'
MF.PACKED = &H40
MF.PLANAR = &H0
MF.INTERLACED = &H2
MF.PLUSPOLARITY = &H8

'* Video board types *'
BT.YUV411 = 0
BT.RGB16 = 1
BT.YUV211 = 2
BT.YUV422 = 3
BT.RGB24 = 4

'* Bitmap types *'
BM.DIB24 = 0                  '* Windows DIB 24 bit true color *'
BM.DIB8P = 1                  '* Windows DIB 8 bit palettized *'
BM.DIB8G = 2                  '* Windows DIB 8 bit gray-scale *'
BM.DIB4D = 3                  '* Windows DIB 4 bit dithered *'
BM.TRG32 = 4                  '* Targa 32 bit true color *'
BM.TRG24 = 5                  '* Targa 24 bit true color *'
BM.TRG16 = 6                  '* Targa 16 bit true color *'
BM.YUV411 = 7                 '* IBM MMotion format 4:1:1 YUV *'
BM.TIFF8P = 8                 '* TIFF 8 bit palettized *'
BM.TIFF8G = 9                 '* TIFF 8 bit gray-scale *'
BM.TIFF4D = 10                '* TIFF 4 bit dithered *'
BM.PCX8P = 11                 '* PCX 8 bit palettized *'
BM.PCX8G = 12                 '* PCX 8 bit gray-scale *'
BM.PCX4D = 13                 '* PCX 4 bit dithered *'
BM.GIF8P = 14                 '* GIF 8 bit palettized *'
BM.GIF8G = 15                 '* GIF 8 bit gray-scale *'
BM.JPEG = 16                  '* JPEG compressed 4:2:2 YUV *'

NO.COLOR.SETTINGS = 8         '* Available color controls *'
NO.COLOR.CONTROLS = 7         '* Actual number of color controls *'
NO.SKEW.FACTORS = 7           '* Number of skew factors *'
NO.VIDEO.MODES = 16           '* Number of video modes supported *'
NO.PHIXEL.CHIPS = 4           '* Number of Phixel chips supported *'

'* Configuration flags *'
CF.PAL = 0                    '* Input format is PAL *'
CF.NTSC = 1                   '* Input format is NTSC *'
```

```
CF.HASPLL = 2                         '* Board has a phase-locked loop *'
CF.REPLICATE = REPLICATE.ON           '* Replicate field feature enabled *'

'* Skew adjustment IDs *'
SF.DISPWINSKEWX = 0                   '* Display Window X Skew *'
SF.DISPWINSKEWY = 1                   '* Display Window Y Skew *'
SF.DISPADDRSKEWX = 2                  '* Display Address X Skew *'
SF.DISPADDRSKEWY = 3                  '* Display Address Y Skew *'
SF.SHIFTCLOCKSTART = 4                '* Shift Clock Start *'
SF.PALETTESKEW = 5                    '* Palette Skew *'
SF.PLLDIVISOR = 6                     '* Phase Lock Loop Divisor *'

'* Skew adjustment ranges *'
DISPWINSKEWXRANGE = &H7FF             '* Display Window X Skew Range *'
DISPWINSKEWYRANGE = &H3FF             '* Display Window Y Skew Range *'
DISPADDRSKEWXRANGE = &H3FF            '* Display Address X Skew Range *'
DISPADDRSKEWYRANGE = &H1FF            '* Display Address Y Skew Range *'
SHIFTCLOCKSTARTRANGE = &H7F           '* Shift Clock Start Range *'
PALETTESKEWRANGE = &H3                '* Palette Skew Range *'
PLLDIVISORRANGE = &H3FF               '* Phase Lock Loop Divisor Range *'

MAX.PALETTE.ENTRIES = 236             '* Windows reserves 20 of the 256 colors *'
pf.pALFILE = &H4000                   '* Output a palette file *'
iColor = 1'                           '* Color index *'
DEFINT A-Y                            '

ESC.KEY = &H27
Q.KEY = &H71                          '* Extended scan code for Q *'
P.KEY = &H70                          '* Extended scan code for P *'
C.KEY = &H63                          '* Extended scan code for C *'
F.KEY = &H66                          '* Extended scan code for F *'
S.KEY = &H73                          '* Extended scan code for S *'
SCREENWIDTH = 640                     '* Screen width *'
SCREENHEIGHT = 480                    '* Screen height *'
FITWINDOW = 1                         '* Fit video flag *'
FULLSIZEWINDOW = 0
MOVENO = 100                          '* Moving count *'
XINC = 4                              '* X increment *'
YINC = 3                              '* Y increment *'
SIZE.X.INC = 4
SIZE.Y.INC = 3
```

程序 8.5 MS BASIC 7.1 PCVIDEO.QBI 内包文件

对于 VISUAL BASIC FOR DOS，建成快速库的方法是：

```
LINK/Q/NOD/CO PCVIDEOC PCVIDEO FRAMEGRB CONVERT,
PCVIDEO. QLB,,VBDOSQLB. LIB
LIB PCVIDEO. LIB PCVIDEOC. OBJ+PCVIDEO. OBJ+CONVERT. OBJ+FRAMEGRB. OBJ
```

由于 VISUAL BASIC FOR DOS 和 C 语言一样，支持"_"作为变量或子程序名，因此，它的内包文件 EXAMPLE. BI 和源程序更接近于 C 语言，其余操作和 MS BASIC 7.1 基本相同。

应该告知读者，我们在建以上两种 BASIC 的快速库时，都遇到同一类型困难：以 MS C/C++7.0 为例，我们把 VIDEO PLUS 所需目标文件都拷贝到 BC7\子目下，然后用如下批文件来建 PCVIDEO. QLB：

```
erase pcvideo. lib
LIB PCVIDEO. lib +pcvideoc+framegrb+pcvideo+convert+
palette+platform+LIB\qbx. lib
link/Q/NOD/CO pcvideo. lib,pcvideo. qlb,PCVQLB. ERR, LIB\qbxqlb. lib
```

这是因为 PCVIDEO 的编译成功的目标文件 *. OBJ 都必需和 MLIBCE. LIB 相联结才能工作，而这些库和 MS BASIC 7.0 中的 QBXQLB. LIB 或 VISUAL BASIC FOR DOS 中的 VBDOSQLB. LIB 都多少有些矛盾. 也就是说，PCVIDEO 的源程序编写时，没有考虑到它们的独立完整性，过分的依赖了 MS C 语言库函数。尽关 MS C/C++ 7.0，MS BASIC 7.1 和 VISUAL BASIC FOR DOS 都是 MICROSOFT 公司的软件产品，它们的库之间并不完全兼容。而快速库的子程序必需是完整的。如果要修改以上 PCVIDEO 源程序，补充全所缺的原来在 SLICE. LIB 和 OLDNAMES. LIB 中的子程序，则工作量太大不如重编。由于时间关系，我们努力到此为止. 希望有兴趣的读者不吝指教。

PCVIDEO. LIB 可以增删. VIDEOPLUS 就是它的改进版。它的组成源程序又多了 COMPRESS. ASM 和 PLATFORM. ASM，原有四个程序篇幅也有所增加或改写. 内包文件内容也增加了。对应的在 C 和 MASM 中 MAK 文件(两个同名文件 PCVIDEO)的内容和此处有所不同. 无论如何，以上方法原则上仍是可用的。

PCVIDEO 的源程序也可适当简化。例如删去有关 WINDOWS 的应用部分。有几种可能性的，例如不同版本号，不同图像文件格式可改成只选一种可能。五十个功能库函数中外部可以不用，内部也不用的可删除. 这样的 PCVIDEO. LIB 可以作得很小。如果从零开始编 GRAND VIDEO 的程序，作者认为，功能库中不少对运行速度要求不高的部分，例如初始和结束，有些寄存器操作，完全可用 BASIC 语言来编. 这样程序会小得多。如果不追求完整的帧之间的联结，只用最简单的 BASIC 语言也是可以发挥 GRAND VIDEO 的全部功能的。

GRAND VIDEO 的最佳的编程方案应该是 VISUAL BASIC FOR DOS 专业版和少量必要的 MASM 的子程序有机结合。这时 BASIC 本身支持窗口，鼠标，菜单和数据库，它胜任与 GRAND VIDEO 硬件无关的各种任务。混合语言编程的快速库完成与硬件有关的操作或实时图像压缩及变换。非矩形窗口需要利用色键和作图功能。应该指出，VISUAL BASIC 只有在用 SCREEN. HIDE 清除所有的窗体后，才能利用它全部的作图功能。

Document generated by Anna’s Archive around 2023-2024 as part of the DuXiu collection (https://annas-blog.org/duxiu-exclusive.html).

Images have been losslessly embedded. Information about the original file can be found in PDF attachments. Some stats (more in the PDF attachments):
{
"filename": "6KeG6aKR5aSa5aqS5L2T56Gs5Lu25oqA5pyv5LiO6L2v5Lu257yW56iLXzEyNzU3MDA1LnppcA==",
"filename_decoded": "\u89c6\u9891\u591a\u5a92\u4f53\u786c\u4ef6\u6280\u672f\u4e0e\u8f6f\u4ef6\u7f16\u7a0b_12757005.zip",
"filesize": 44495622,
"md5": "507e79455e383b96d5bd8795eca730de",
"header_md5": "a7f68a4b9f901f41e82521ee1a842178",
"sha1": "270f6ba6730a5a5a3797a3e88108ac76e659606b",
"sha256": "c8507606f798d3505cdb552d8df683f3fe52691d7b6edec2d76f80bcfc9aad2a",
"crc32": 3070962573,
"zip_password": "",
"uncompressed_size": 48466499,
"pdg_dir_name": "\u2569\u2559\u255e\u2561\u2562\u03b1\u251c\u255c\u2560\u03c3\u2559\u2593\u255d\u25a0\u255d\u255d\u2569\u2321\u2559\u03b4\u255a\u03c6\u255d\u25a0\u2592\u03b1\u2502\u2560_12757005",
"pdg_main_pages_found": 254,
"pdg_main_pages_max": 254,
"total_pages": 260,
"total_pixels": 1546396152,
"pdf_generation_missing_pages": false
}